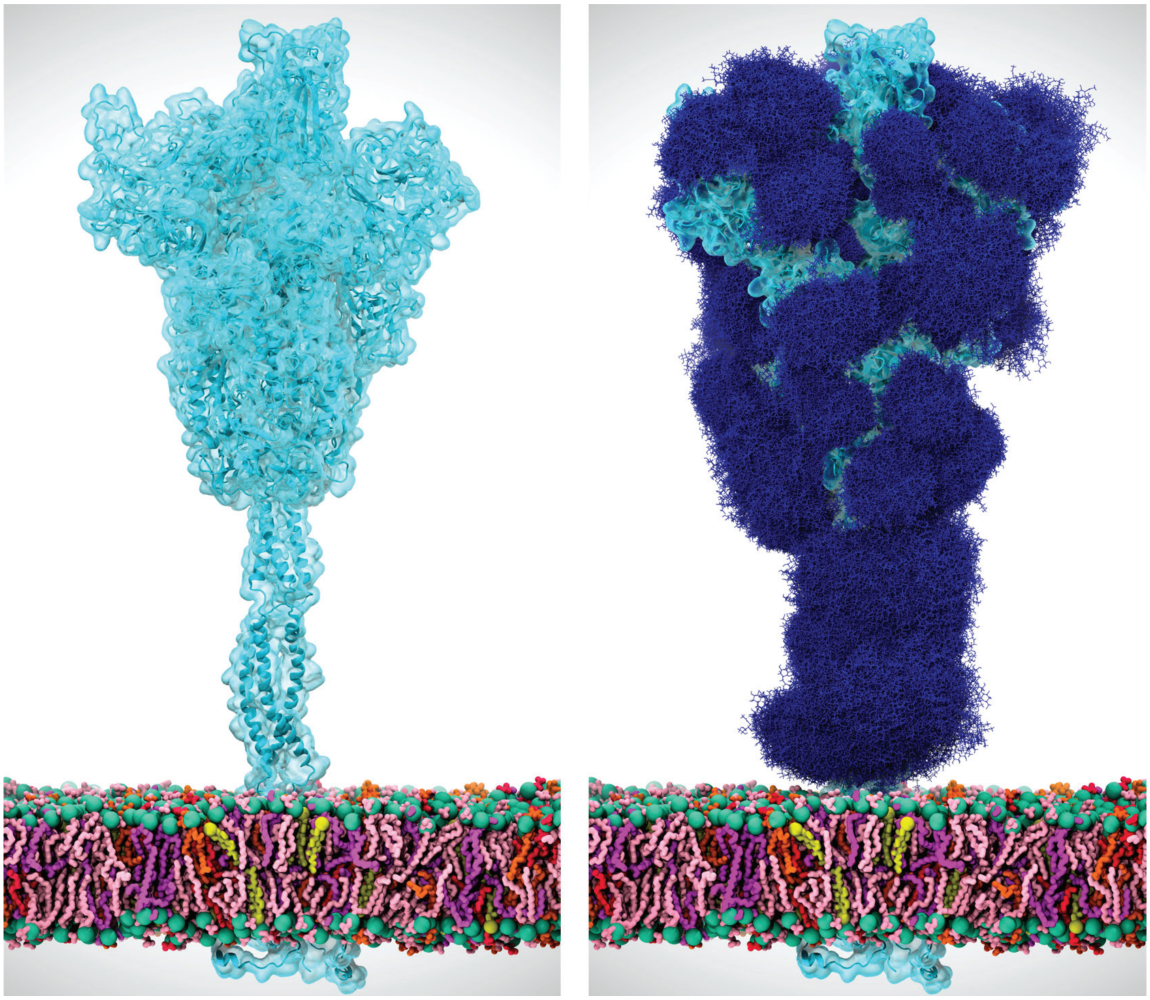
Computing in SCIENCE & ENGINEERING

VOLUME 22, NUMBER 6

NOVEMBER/DECEMBER 2020



Computational Science in the Battle Against COVID-19

IEEE

IEEE COMPUTER SOCIETY

www.computer.org/cise

Thank You to Our Sponsors!

CVPR 2020—the premier annual conference on computer vision and pattern recognition—virtually delivered more than 5,000 first-class papers, keynotes, sessions, workshops, and tutorials to an audience of 7,600 from all over the world!

We are so grateful to our wonderful volunteers and our amazing lineup of sponsors for their continuing support.

Champion

Alibaba Group 阿里巴巴集团 · amazon | science · appen

MEGVII 旷视 · Microsoft · NAVER | LINE | LABS NAVER LABS

Qualcomm · SuperAnnotate

Supporter

AI.REVERIE · algolux · Bai du 百度

EXXACT · FACEBOOK · FUTUREWEI Technologies · Google

HITACHI Inspire the Next · IBM · iMerit · inspur

intel REALSENSE TECHNOLOGY · NSA WHERE INTELLIGENCE GOES TO WORK® IntelligenceCareers.gov/NSA · NVIDIA · OAK RIDGE National Laboratory

Uber · WAYMO

cvpr20.com

IEEE COMPUTER SOCIETY · CVF

Editorial: Unless otherwise stated, bylined articles, as well as product and service descriptions, reflect the author's or firm's opinion. Inclusion in *Computing in Science & Engineering* does not necessarily constitute endorsement by IEEE or the IEEE Computer Society. All submissions are subject to editing for style, clarity, and length. IEEE prohibits discrimination, harassment, and bullying. For more information, https://www.ieee.org/about/corporate/governance/p9-26.html. **Circulation:** *Computing in Science & Engineering* (ISSN 1521-9615) is published bimonthly by the IEEE Computer Society. IEEE Headquarters, Three Park Ave., 17th Floor, New York, NY 10016-5997 USA; IEEE Computer Society Publications Office, 10662 Los Vaqueros Cir., Los Alamitos, CA 90720 USA, phone +1 714 821 8380; IEEE Computer Society Headquarters, 2001 L St., Ste. 700, Washington, D.C. 20036 USA. Subscribe to *Computing in Science & Engineering* by visiting www.computer.org/cise. **Reuse Rights and Reprint Permissions:** Educational or personal use of this material is permitted without fee, provided such use: 1) is not made for profit; 2) includes this notice and a full citation to the original work on the first page of the copy; and 3) does not imply IEEE endorsement of any third-party products or services. Authors and their companies are permitted to post the accepted version of IEEE-copyrighted material on their own web servers without permission, provided that the IEEE copyright notice and a full citation to the original work appear on the first screen of the posted copy. An accepted manuscript is a version that has been revised by the author to incorporate review suggestions, but not the published version with copyediting, proofreading, and formatting added by IEEE. For more information, please go to: https://www.ieee.org/publications/rights/author-posting-policy.html. Permission to reprint/republish this material for commercial, advertising, or promotional purposes or for creating new collective works for resale or redistribution must be obtained from IEEE by writing to the IEEE Intellectual Property Rights Office, 445 Hoes Lane, Piscataway, NJ 08854-4141 USA or pubspermissions@ieee.org. Copyright © 2020 IEEE. All rights reserved. **Abstracting and Library Use:** Abstracting is permitted with credit to the source. Libraries are permitted to photocopy for private use of patrons, provided the per-copy fee indicated in the code at the bottom of the first page is paid through the Copyright Clearance Center, 222 Rosewood Dr., Danvers, MA 01923 USA. **Postmaster:** Send undelivered copies and address changes to Computing in Science & Engineering, 445 Hoes Ln., Piscataway, NJ 08855 USA. Periodicals postage paid at New York, NY, and at additional mailing offices. Canadian GST #125634188. Canada Post Corporation (Canadian distribution) publications mail agreement number 40013885. Return undeliverable Canadian addresses to PO Box 122, Niagara Falls, ON L2E 6S8 Canada. Printed in the USA.

Contents

November/December 2020
Volume 22 Number 6

www.computer.org/cise

ISSN: 1521-9615

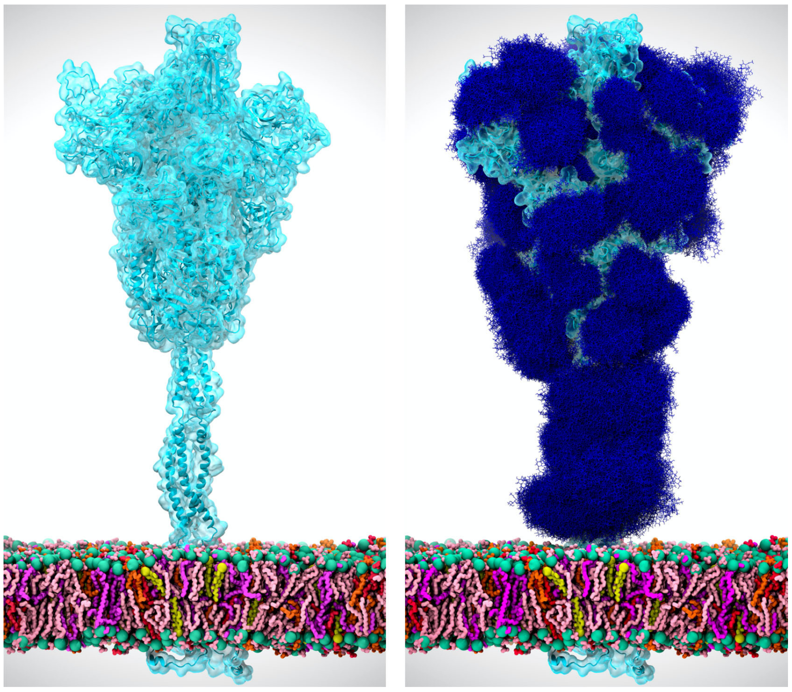


Figure 1 from "Biomolecular Simulations in the Time of COVID-19, and After," by Amara and Mulholland, p. 30

COLUMNS AND DEPARTMENTS

Computing in SCIENCE & ENGINEERING

PURPOSE: The IEEE Computer Society is the world's largest association of computing professionals and is the leading provider of technical information in the field.

MEMBERSHIP: Members receive the monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.

COMPUTER SOCIETY WEBSITE: www.computer.org

OMBUDSMAN: Direct unresolved complaints to ombudsman@computer.org.

CHAPTERS: Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

AVAILABLE INFORMATION: To check membership status, report an address change, or obtain more information on any of the following, email Customer Service at help@computer.org or call +1 714 821 8380 (international) or our toll-free number, +1 800 272 6657 (US):

- Membership applications
- Publications catalog
- Draft standards and order forms
- Technical committee list
- Technical committee application
- Chapter start-up procedures
- Student scholarship information
- Volunteer leaders/staff directory
- IEEE senior member grade application (requires 10 years practice and significant performance in five of those 10)

PUBLICATIONS AND ACTIVITIES

***Computer*:** The flagship publication of the IEEE Computer Society, *Computer* publishes peer-reviewed technical content that covers all aspects of computer science, computer engineering, technology, and applications.

Periodicals: The society publishes 12 magazines and 18 journals. Refer to membership application or request information as noted above.

Conference Proceedings & Books: Conference Publishing Services publishes more than 275 titles every year.

Standards Working Groups: More than 150 groups produce IEEE standards used throughout the world.

Technical Committees: TCs provide professional interaction in more than 30 technical areas and directly influence computer engineering conferences and publications.

Conferences/Education: The society holds about 200 conferences each year and sponsors many educational activities, including computing science accreditation.

Certifications: The society offers three software developer credentials. For more information, visit www.computer.org/certification.

BOARD OF GOVERNORS MEETING

24 – 25 September 2020 in McLean, Virginia, USA

revised 1 May 2020

COMPUTER SOCIETY OFFICES

Washington, D.C.: 2001 L St., Ste. 700, Washington, D.C. 20036-4928; **Phone:** +1 202 371 0101; **Fax:** +1 202 728 9614; **Email:** help@computer.org
Los Alamitos: 10662 Los Vaqueros Cir., Los Alamitos, CA 90720; **Phone:** +1 714 821 8380; **Email:** help@computer.org

MEMBERSHIP & PUBLICATION ORDERS

Phone: +1 800 678 4333; Fax: +1 714 821 4641; Email: help@computer.org

Computational Science and Engineering in 2020

Lorena A. Barba
The George Washington University

In 1995, just one year after the founding of *IEEE Computational Science & Engineering*—the precursor of CiSE—founding Editor-in-Chief Ahmed H. Sameh wrote that the board's goal was "making this magazine the flagship of all aspects of computational science and engineering."[1] What was it like at the time? 1995 marks the beginning of the dot-com boom. That year, Amazon and eBay first opened their digital doors, the Intel Pentium Pro was released, IBM unveiled Deep Blue, and the NumPy library for array computing in Python was first introduced. It was also the year that Peter Norvig and Stuart Russell published their classic textbook on artificial intelligence.[2] The University of Texas at Austin established its program in computational and applied mathematics, which J. Tinsley Oden announced in *CS&E*.[3] The dozen tenure-track positions opened with the new program and institute were housed in the departments of mathematics, computer science, engineering mechanics, aerospace engineering, and other engineering specialties. It was conceived as a broadly interdisciplinary program, and that has been the hallmark of computational science and engineering across time.

The interdisciplinary essence of computational science has placed the field in the bewildering position of being central to vast swaths of new discoveries while also lacking recognition and support to flourish. A PITAC report in 2005 declared that "much of the promise of computational science remains unrealized due to inefficiencies within the R&D infrastructure and lack of strategic planning and execution."[4] It called for supporting computational science as a "national imperative for research and education in the 21st century." The core hurdle was and is discipline-based research silos interfering with the integration of diverse skills and knowledge sets needed for advanced computational research. Quoting the report, "inadequate and outmoded educational structures within academia [. . .] leave computational science students to flounder amid competing departments." This was ten years after the founding of UT Austin's new institute (now called the Oden Institute for Computational Engineering and Sciences, https://www.oden.utexas.edu) and ten years before the Department of Computational Mathematics, Science, and Engineering was established at Michigan State University (https://cmse.msu.edu). It was also the year when we faced up to the reality that the "free lunch" of continued increases in processor performance was over.[5] Clock speeds reached the 3-GHz level—where they remain to this day—and major manufacturers turned to multicore architectures. The clock race ended and a mainstream 4-GHz processor never landed. "Concurrency really is hard," Herb Sutter

Digital Object Identifier 10.1109/MCSE.2020.3027933
Date of current version 9 October 2020.

wrote, and so it became harder to train computational scientists.

The landscape of computing soon became even more thorny, as programmable general-purpose GPUs hit the scene. In 2007, NVIDIA released the CUDA software development kit and the first graphics processors designed for computing. While the trend then called "GPGPU" picked up speed, the first highly influential applications of GPUs for deep learning were published, and a new era unleashed.[6] Many of us went on long expeditions of adapting our best-loved algorithms for solving physical models formulated as differential equations to the new architecture and programming models. Some were decidedly successful, like those in the molecular modeling and quantum chemistry communities, for example. But for most it has either been a grind, or a trend to be avoided. In the last ten years, to cap all this, computational science was inundated with new approaches like machine learning and data-based models. Now, the trendy exploit is to try "physics-informed" neural networks in applications.

Computational science and engineering for many years were dominated by physical simulations based on mathematical modeling and numerical analysis. This is reflected in the articles CiSE published, on topics like computational electromagnetics, molecular dynamics for modeling macroscopic material properties, simulations of the earth's mantle convection-driven flow, quantum and molecular mechanics combined for chemistry applications, and geographic environmental modeling (all in 1995). Scope broadened to areas like computation in medicine (2000, issue 5), data mining (2002, issue 4), and cloud computing (2009, issue 4). Notably, two special-issue themes were harbingers of topics trending today: Python for scientific computing (2007, issue 3) and reproducible research (2009, issue 1). In the first, some of CiSE's most highly cited articles appear, including the Matplotlib paper[7] with more than 11 400 citations indexed by Google Scholar as of this writing. The theme reappeared in 2011 (issue 2), with an article on the NumPy array structure that has amassed more than 6000 citations, so far. In more recent years, it has been hard to resist the onslaught of interest in machine learning and related topics. CiSE first covered machine learning in 2013 (issue 5), highlighting applications like materials sciences and climate informatics. The pull of computer science perspectives, versus computational science, began to sway the content before long. Do we need to reclaim focus?

As interim Editor-in-Chief for the year 2020, and now appointed the EiC for the three-year period starting 2021, I have been reflecting on what focus I wish to bring to CiSE. Certainly, the early concerns about the hurdles of interdisciplinarity and the educational needs of the new generations of computational scientists are high on my list. Revisiting the role of the Python ecosystem in computational science seems timely now, with the recent convergence with GPU computing via projects to develop open-source libraries giving access to GPU devices from Python programs. In general, I aim for a heightened emphasis on open-source research software, combined with attention for transparency and reproducibility. I was a member of the study committee of the National Academies that produced the consensus report on Replicability and Reproducibility in Science last year,[8] and it has been a long-time area of focus in my own research. Even if CiSE featured the topic in 2009 and 2012, it has trended mainstream only recently—it is time to integrate those considerations firmly into the publication process.

In 2017, Reproducible Research became a new peer-reviewed track for CiSE, with outgoing EiC George Thiruvathukal and myself as coeditors. There, we have spearheaded practices enhancing transparency and supporting open science, like asking reviewers if they want to participate in open peer review (open identity, and open report) and asking authors to post a preprint of their manuscript. Reviewers who opt-in have deposited their review reports in services like Figshare or Authorea (https://www.authorea.com/inst/18992), and authors cite the review reports with the preprint identifiers, acknowledging the reviewer contributions. This open-review process works within the existing systems, and was approved by the Computer Society VP of Publications in April 2018. I am interested in expanding this model to other submissions, perhaps some special issues.

As to topical focus, my goal is to realign CiSE with its origins of publishing *"articles that help define the field of computational science and engineering, emphasizing significant computational contributions to science and engineering discipline"* (citing first EIC Ahmed Sameh's launch editorial).

A higher aspiration was alluded to by past EIC Francis Sullivan on occasion of the magazine's 20th anniversary, to be "a broad-based publication. . . about the fundamental role of computation in all aspects of human civilization."[9]

In this issue, CiSE tackles the historic menace to our way of life that is the COVID-19 pandemic, and the role of computing and data science in our battle against it. This is a milestone for CiSE, and it was made possible by the formidable efforts of editors, authors, and reviewers to produce a high-quality issue with a breakneck timeline. The Computer Society recognizes this effort and the importance of the theme by making all these articles free to read on the Computer Society Digital Library for six months. I hope they will be read widely and illuminate the power of computing as a scientific tool in the service of society.

To close this editorial, I would like to recognize the new editors who accepted my invitation to join the board this year:

1) **Anne Elster** (Novel Architectures), Professor of Computer Science and HPC, Norwegian University of Science and Technology.
2) **Dirk Colby** (Education), Director of HPC Studies, Computational Mathematics, Science and Engineering, Michigan State University.
3) **Sharon Broude Geva** (Education), Director of Advanced Research Computing, University of Michigan.
4) **Michela Taufer**, Dongarra Professor in High Performance Computing, University of Tennessee Knoxville.
5) **Karla Morris** (Software Engineering), Principal Member of Technical Staff at Sandia National Laboratories.
6) **Ilkay Altintas** (Data Track), Chief Data Science Officer at the San Diego Supercomputer Center, UCSD.
7) **Anna-Karin Tornberg**, Professor of Numerical Analysis, Sweedish Royal Institute of Technology.
8) **Ewa Deelman**, Principal Scientist, USC Information Sciences Institute, University of Southern California.

Starting January 2021, we welcome:

1) **Kelly Gaither** (Visualization Corner), Director of Visualization at the Texas Advanced Computing Center (TACC), University of Texas.
2) **Karen Willcox**, Professor of Aerospace Engineering and Engineering Mechanics and Director of the Oden Institute for Computational Engineering and Sciences, University of Texas.
3) **John M. Shalf** (Leadership Computing), Department Head for Computer Science, Lawrence Berkeley National Laboratory.
4) **Kathryn Mohror** (Leadership Computing), Computer Scientist at Lawrence Livermore National Laboratory.

I look forward to working with all of them, and our continuing editors, to realize the aspirations for CiSE to be the flagship of all aspects of computational science and engineering.

REFERENCES

1. A. Sameh, " Looking back, looking forward: Evaluation and vision," *IEEE Comput. Sci. Eng.*, vol. 2, no. 2, p. 1, Summer 1995, doi: 10.1109/MCSE.1995.10010.
2. P. Norvig and S. Russell, *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1995, doi: 10.1109/99.388942.
3. J. T. Oden, "UT Austin establishes computational and applied mathematics program," *IEEE Comput. Sci. Eng.*, vol. 2, no. 2, pp. 7–9, Summer 1995, doi: 10.1109/99.388942.
4. R. Bajcsy *et al.*, "Computational science: ensuring America's competitiveness," President's Inf. Technol. Advisory Committee, Arlington, VA, USA, 2005. [Online]. Available: https://apps.dtic.mil/sti/pdfs/ADA462840.pdf
5. H. Sutter, "The free lunch is over: A fundamental turn toward concurrency in software," *Dr. Dobb's J.*, vol. 30, no. 3, pp. 202–210, 2005.
6. R. Raina, A. Madhavan, and A. Y. Ng, "Large-scale deep unsupervised learning using graphics processors," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 873–880, doi: 10.1145/1553374.1553486.
7. J. D. Hunter, "Matplotlib: A 2D graphics environment," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, May/Jun. 2007, doi: 10.1109/MCSE.2007.55.
8. National Academies of Sciences, Engineering, and Medicine, *Reproducibility and Replicability in Science*. Washington, DC, USA: Nat. Acad. Press, 2019, doi: 10.17226/25303.
9. F. Sullivan, N. Chonacky, I. Beichl, and G. K. Thiruvathukal, "Former CiSE EICs reflect on the magazine's 20th anniversary," *Comput. Sci. Eng.*, vol. 20, no. 1, pp. 3–7, Jan./Feb. 2018, doi: 10.1109/MCSE.2018.011111118.

stay connected.

Computational Science in the Battle Against COVID-19

Fernanda Foertter
BioTeam

Kelly Gaither
Texas Advanced Computing Center, UT Austin

Konrad Hinsen
CNRS Orléans

John West
Texas Advanced Computing Center, UT Austin

IT IS WITH a sense of urgency that we present this special issue of CiSE on the role of computing in battling the COVID-19 pandemic. In early January 2020, scientists identified SARS-CoV-2 as the cause of a cluster of pneumonia cases in Wuhan City, Hubei, China. The first official death from COVID-19 was reported by China on January 11, and the first death outside of China was reported on February 2nd. The numbers change every day, but as of this writing 24M people worldwide have been infected and over 800 000 of them have died.

As a near-universal scientific instrument, computing is a versatile and effective tool in many aspects of the pandemic response. In this special issue, we have gathered a set of five invited and peer-reviewed papers that we hope will illuminate several of the ways in which computing is playing an important role in response to the devastation of COVID-19.

The first three papers provide a view of computational activities that are aimed at trying to better understand how the virus works, research that contributes to the search for vaccines, and more effective pharmaceutical therapy. Perilla *et al.* discuss the use of molecular simulations to gain insights into key tdeterminants of infection by characterizing the mechanism by which enveloped viruses, such as SARS-CoV-2, infect cells. Li *et al.* are also involved in the process of understanding infection, using computational simulation to study the features of the interfaces of the so-called spike protein of SARS-CoV-2, which is able to bind to human Angiotensin-Converting Enzyme 2 (ACE2) initializing the entry to the host cell. Amaro *et al.* step back from the details of specific molecular simulations and examine the evolution of the state of practice in biomolecular simulation during the time of COVID-19. Disruptions in established practices include broader adoption of preprint servers, more open sharing of methods, models, and data, and streamlined processes for accessing critical computing resources quickly. The article discusses the interplay of all of these factors and how they influence biomolecular simulations in the fight against SARS-CoV-2.

Computing also plays a role in the design of more effective near-term therapeutic interventions for patients suffering from the worst effects of the disease. Randles *et al.* describe an airflow simulation developed to help address the need to rapidly expand ventilator capacity for hospitalized patients by splitting ventilators between two or more patients with differing respiratory physiologies.

Digital Object Identifier 10.1109/MCSE.2020.3025398
Date of current version 9 October 2020.

Finally, computational models provide important information to health officials and decision-makers as they formulate and implement a public policy response to the pandemic. Reinert *et al*. address the development of a visual analytics application for supporting pandemic preparedness through a tightly coupled epidemiological model and interactive interface.

This special issue also contains several department contributions related to COVID-19. The Visualization Corner in this issue highlights several of the many different visualizations that have played a role in understanding the science and policy surrounding the pandemic. In part one of a two-article series, the Leadership Computing department describes some of the projects that are taking advantage of the largest supercomputing systems in the U.S. computing infrastructure. The Diversity and Inclusion department takes a look at how the pandemic has had a disproportionate impact on members of groups traditionally underrepresented in computing. Finally, in the Education department, the authors explore the impact of the global pandemic on the online course "Problem Solving Using Computational Thinking," with a particular focus on the topics learners chose for their final projects.

Given the urgency of addressing the pandemic and the role of publications in disseminating knowledge, we felt it was especially important to get this issue out in the same year the pandemic started. Of course, that meant a tremendous effort from authors and reviewers to write and review contributions in an incredibly compressed timeline while the pandemic was still ongoing. We are very grateful for their work. The authors are acknowledged by inclusion of their work in this issue, but we want to make a special gesture to recognize and thank the reviewers for their all-too-often under-appreciated contribution to the health of the academic community: Peter J. Bond (Agency for Science, Technology and Research, A*STAR), Min Chen (Oxford University), Marco De Vivo (Italian Institute of Technology), David Emerson (Daresbury Laboratory, U.K. Science and Technology Facilities Council), Thomas Ertl (Universität Stuttgart), Sergei Grudinin (Inria), Michael Hagan (Brandeis University), Yi He (University of New Mexico), Chris North (Virginia Tech), Abhishek Singharoy (Arizona State University), Tamar Schlick (New York University), Luba Tchertanov (École normale supérieure Paris-Saclay), Shaolei Teng (Howard University), Zhicheng Wang (Brown University), Gerhard Wellein (Friedrich-Alexander-Universität Erlangen-Nürnberg), and Shan Zhao (University of Alabama).

Fernanda Foertter is a Senior Scientific Consultant at BioTeam, working on helping organizations with artificial intelligence, machine learning, and technical aspects of doing data science such as methods, data integration, system integration, infrastructure, and scaling. She is also interested in exploring data curation, metadata, processes, and organizational collaborations since data science often uses data from multiple parts of an organization or system. Contact her at ffoertter@bioteam.net.

Kelly Gaither is the Director of Health Analytics at the Texas Advanced Computing Center and Associate Professor, Department of Women's Health at the Dell Medical School, UT Austin. She received the bachelor's and master's degrees in computer science from Texas A&M University in 1988 and 1992, respectively, and the Ph.D. degree in computational engineering from Mississippi State University in May 2000. She has published in a wide variety of venues and her expertise is in visualization, large data analytics, and computational science applications. Contact her at kelly@tacc.utexas.edu.

Konrad Hinsen is a researcher at the Centre National de la Recherche Scientifique who works at the Centre de Biophysique Moléculaire in Orléans and as an associate researcher at the Synchrotron SOLEIL. He received the Ph.D. degree in theoretical physics from RWTH Aachen University, and his current fields of research are the structure and dynamics of proteins, using molecular simulation and statistical physics and the methodology of computational science, in particular concerning reproducibility and verifiability. Contact him at konrad.hinsen@cnrs.fr.

John West is Director of Strategic Initiatives at the Texas Advanced Computing Center, part of the University of Texas at Austin in the United States. TACC provides HPC services and expertise to the open science community and is the largest provider of HPC resources in the NSF XSEDE community. Prior to joining TACC, he was Director of the USA Department of Defense High Performance Computing Modernization Program and held a number of positions in private industry and the federal government providing supercomputing resources and expertise for research and development missions. Contact him at john@tacc.utexas.edu.

Scalable Analysis of Authentic Viral Envelopes on FRONTERA

Fabio González-Arias
University of Delaware

Tyler Reddy
Los Alamos National Laboratory

John E. Stone
University of Illinois at Urbana-Champaign

Jodi A. Hadden-Perilla
University of Delaware

Juan R. Perilla
University of Delaware

***Abstract*—Enveloped viruses, such as SARS-CoV-2, infect cells via fusion of their envelope with the host membrane. By employing molecular simulations to characterize viral envelopes, researchers can gain insights into key determinants of infection. Here, the Frontera supercomputer is leveraged for large-scale modeling and analysis of authentic viral envelopes, whose lipid compositions are complex and realistic. Visual Molecular Dynamics (VMD) with support for MPI is employed, overcoming previous computational limitations and enabling investigation into virus biology at an unprecedented scale. The techniques applied here to an authentic HIV-1 envelope at two levels of spatial resolution (29 million particles and 280 million atoms) are broadly applicable to the study of other viruses. The authors are actively employing these techniques to develop and characterize an authentic SARS-CoV-2 envelope. A general framework for carrying out scalable analysis of simulation trajectories on Frontera is presented, expanding the utility of the machine in humanity's ongoing fight against infectious diseases.**

Viruses are pathogens that cause infectious diseases in living organisms. Because of their lack of metabolic capabilities, viruses require the molecular machinery of a host cell to replicate. Virus particles, referred to as virions, exhibit chemical and structural diversity across families. The detailed architecture of a virus determines its fitness, mechanism of infection and replication, and host cell tropism.

Digital Object Identifier 10.1109/MCSE.2020.3020508
Date of publication 31 August 2020; date of current version 9 October 2020.

The most basic virions consist of genomic material protected by a protein shell, called a capsid. The viral genome contains the blueprint for synthesis of the macromolecular components that will form progeny virions. Depending on the virus, the genome may be encoded as RNA or as DNA. More complex virus structures exhibit an exterior membrane, called an envelope, composed of a lipid bilayer. Fusion of the viral envelope with the membrane of the host cell initiates infection. Enveloped viruses typically incorporate surface glycoproteins that interact with host cell receptors to mediate adhesion and facilitate membrane fusion. The composition of the viral envelope, along with the specificity of the adhesion proteins, contributes to the recognition, attachment, and fusion of the pathogen with its host. Other membrane-embedded structures that may be displayed by a virion include viral ion channels, called viroporins, proteins that provide particle scaffolding or assembly support, or proteins that participate in the release of progeny virions.

Enveloped viruses, such as influenza A, Ebola, and HIV-1, account for numerous cases of infection and death in humans each year. SARS-CoV-2, the novel coronavirus that causes COVID-19, is also enveloped. The SARS-CoV-2 virion incorporates a class-I fusion glycoprotein called the spike (S), a purported viroporin known as the envelope (E) protein, and a structurally essential integral membrane (M) protein. While each of these components plays a key role in the viral life cycle, it is ultimately the envelope that enables each to carry out its function successfully. As the envelope is derived from the host cell membrane, particularly from the plasma membrane or organelle in which the virion assembles, its lipid composition may be highly complex. Furthermore, the envelope may be asymmetric, or present different lipid compositions across the inner versus outer leaflets of the bilayer; membrane asymmetry is crucial to the function of most organelles and may likewise be important in the assembly or infection processes of viruses.

Detailed characterization of viruses is critical to the development of new antiviral therapies. Computational studies now routinely contribute to the foundation of basic science that drives innovation in pharmacy, medicine, and the management of public health. Notably, molecular dynamics (MD) simulations have emerged as a powerful tool to investigate viruses. Following advances in high-performance computing, MD simulations can now be applied to elucidate chemical–physical properties of large, biologically relevant virus structures, revealing insights into their mechanisms of infection and replication that are inaccessible to experiments. In recent years, MD simulations of intact enveloped virions, including influenza A, dengue, and an immature HIV-1 particle, have been reported.[1–3] State-of-the-art lipidomics profiling of viruses has enabled some of these models to contain realistic lipid compositions, leading to the first computational studies of authentic viral envelopes.[1]

Here, next-generation modeling and analysis of authentic viral envelopes is discussed, leveraging the resources of the leadership-class Frontera supercomputer.[4] A framework for deploying large-scale analysis of MD simulation trajectories on Frontera is presented. The model system used for demonstration is an authentic HIV-1 envelope, measuring 150 nm in diameter, investigated at two levels of spatial resolution (29 million particles and 280 million atoms). The techniques applied here are broadly applicable to the study of other viruses, including SARS-CoV-2. The authors are actively employing similar techniques on Frontera to develop and characterize an authentic SARS-CoV-2 envelope, which will serve as the foundation for a complete model virion. Given that the described HIV-1 envelope represents an upper size limit for experimentally observed SARS-CoV-2 envelope diameters (75–120 nm), users can expect similar computational requirements and performance outcomes for investigations of SARS-CoV-2 virions on Frontera.

COMPUTATIONAL CHALLENGES

MD simulations of membranes can be carried out at coarse grained (CG) or atomistic levels of detail. CG models employ particles that consolidate groups of atoms. Atomistic models employ discrete particles for each constituent atom, including hydrogen. Due to the size and chemical complexity of authentic viral envelopes, MD simulations can include millions—

even hundreds of millions—of particles, particularly when the physiological solvent environment of the system is taken into account. Moreover, MD simulations of membranes must be performed on timescales of hundreds of nanoseconds to microseconds in order to characterize dynamical properties, such as lipid lateral diffusion and flip-flop between leaflets of the bilayer.

CG models reduce computational expense by eliminating degrees of freedom, enabling the exploration of extended simulation timescales; however, the loss of detail versus atomistic models reduces simulation accuracy. An approach that combines the strengths of both methods involves building and simulating a CG model, which reproduces essential features of the membrane and allows equilibration of lipid species, then backmapping the CG representation to an atomistic model for further study.[5] The backmapping process can be precarious and depends on the molecular mechanics force field and energy minimization scheme to resolve structural artifacts. An alternative approach is the use of all-atom MD simulations to train or calibrate CG conformational dynamics, thereby refining parameterization of the CG membrane model to enhance accuracy. Either way, the incredible number of particle–particle interactions that must be computed to reproduce the behavior of a membrane bilayer over extended simulation timescales, especially for complex, large-scale systems like viral envelopes, presents a significant computational challenge at CG or atomistic resolution.

Furthermore, the amount of data generated by MD simulations of intact viral envelopes is tremendous, both in terms of chemical intricacy and storage footprint. MD simulation trajectories, which may range from tens to hundreds of terabytes in size, fail to provide new information about viruses until they are analyzed in painstaking detail. The magnitude of trajectory files and the sheer numbers of particles that must be tracked during analyses necessitates massively parallel computational solutions. It follows that such large datasets cannot be feasibly transferred to other locations for analysis or visualization, and interaction with the data to yield scientific discoveries must be conducted on the same high-performance computing resource used to run the simulation in the first place.

COMPUTATIONAL SOLUTIONS ON FRONTERA

Frontera is a leadership-class petascale supercomputer, funded by the National Science Foundation and housed at the Texas Advanced Computing Center at the University of Texas at Austin.[4] As of June 2020, it is ranked as the eighth most powerful supercomputer in the world, and the fastest on any university campus. Frontera provides an invaluable computational resource for furthering basic science research of large biological systems, including enveloped viruses.

Frontera comprises multiple computing subsystems. The primary partition is CPU-only and consists of 8008 compute nodes powered by Intel Xeon Platinum 8280 (CLX) processors; each node provides 56 cores (28 cores per socket) and 192 GB of RAM. The large memory partition includes 16 additional compute nodes with 112 cores (28 cores per socket) and memory upgraded to 2.1 TB NVDIMM. Another partition is a hybrid CPU/GPU architecture, consisting of 90 compute nodes powered by Intel Xeon E5-2620 v4 processors and NVIDIA Quadro RTX 5000 GPUs; each node provides 16 cores (8 cores per socket), four GPUs, and 128 GB of RAM. All Frontera nodes contain 240 GB SSDs and are interconnected with Mellanox HDR InfiniBand. The Longhorn subsystem consists of 96 hybrid CPU/GPU compute nodes powered by IBM POWER9 processors and NVIDIA Tesla V100 GPUs; each node provides 40 cores (20 cores per socket), four GPUs, 256 GB of RAM, and 900 GB of local storage. Eight additional nodes have RAM upgraded to 512 GB to support memory-intensive calculations. Longhorn is interconnected with Mellanox EDR InfiniBand.

Frontera has a multitier file system and provides 60 PB of Lustre-based storage shared across nodes. The ability to store massive amounts of data and analyze this data in parallel enables the investigation of biological systems at an unprecedented scale. Researchers gain access, not only to the computational capability to perform MD simulations of millions of particles, but also to the capability to extract more complex and comprehensive information from their simulation trajectories, leading to deeper discoveries relevant to the advancement of human health.

Frontera's Lustre filesystem is crucial for the performance of large-scale biomolecular analysis. Lustre decomposes storage resources among a large number of so-called object store targets (OSTs) that are themselves composed of high-performance RAID arrays. In much the same way that a RAID-0 array can stripe a file over multiple disks, Lustre can similarly stripe files over multiple OSTs. This second level of striping over OSTs is particularly advantageous when working with very large files, since I/O operations at different file offsets can be serviced in parallel by multiple independent OSTs, according to the stripe count and stripe size set by the user, or by system defaults. Figure 1A illustrates how an MD simulation trajectory (and even individual frames) can be striped over OSTs. Figure 1B demonstrates the scalability of multimillion particle biomolecular analysis on Frontera when using the Lustre filesystem.

PARALLEL ANALYSIS WITH VMD

Visual molecular dynamics (VMD) is a widely used software for biomolecular visualization and analysis.[6] Commonly utilized as a desktop application to prepare MD simulations and interact with trajectory data, VMD supports biomolecular systems of up to 2 billion particles, but memory capacity and the performance of I/O, interactive graphics and visualization, and analytical computations determine its usability for such large systems. VMD exploits multicore CPUs, CPU vectorization, and GPU-accelerated computing techniques to achieve high performance for key molecular modeling and visualization tasks. Interactive, exploratory visualizations of large virus structures (several hundred million particles) can be readily performed using GPU-accelerated workstations with large memory capacities, but long-timescale analytical tasks require even greater I/O and computing resources. VMD can also be used *in situ* on massive parallel computers to perform large-scale modeling tasks, exploiting computing and I/O resources that are orders of magnitude greater than those available on even the most powerful desktop workstations. MPI implementations of VMD have already been employed on petascale supercomputers with Lustre filesystems to enable novel investigation of large virus structures, such as the capsid of HIV-1.[2]

To facilitate large-scale molecular modeling pipelines, VMD implements distributed memory message passing with MPI, a built-in parallel work scheduler with dynamic load balancing, and easy to use scripting commands, enabling large-scale

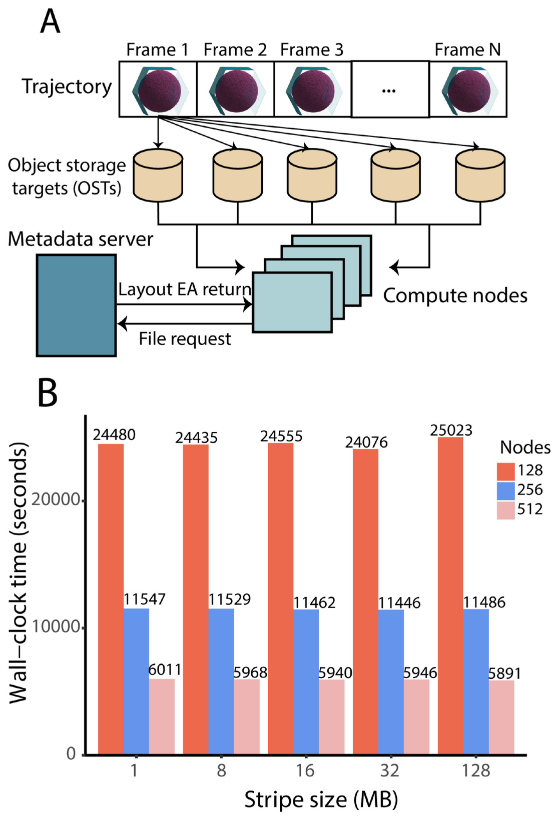


Figure 1. Analysis of large-scale MD simulation trajectories using Lustre filesystem **A** Illustration of trajectory file striping using Lustre filesystem. Each frame on the file is striped across multiple OSTs, where the number of stripes are spawned according to the stripe count and stripe size of the storage file. Upon submission of the job, the compute nodes access a series of frames, from the OSTs, to perform the analysis of the trajectory using parallel I/O. Instructions from the Tcl scripting interface involving reading and writing files are requested through the metadata server (MS), fetching the layout extended attributes and using this information to perform I/O on the file. **B** Wall-clock time for the transverse diffusion analysis of authentic viral envelopes on Frontera, using a stripe count of 16 with different stripe sizes.

parallel execution of molecular visualization and analysis of MD simulation trajectories. VMD's Tcl and Python scripting interfaces provide a user-friendly mechanism to distribute and schedule user-defined work across MPI ranks, to synchronize workers, and to gather results. This approach allows user-written modeling and analysis scripts to be readily adapted from existing scripts and protocols that have been developed previously on local computing resources, allowing robust tools to be deployed on a large-scale, often with few changes.

VMD exploits node-level hardware-optimized CPU and GPU kernels, all of which are also used when running on distributed memory parallel computers. Computationally demanding VMD commands are executed by the fastest available hardware-optimized code path, with fall-back to a general purpose C++ implementation. When run in parallel, the I/O operations of each VMD MPI rank are independent of each other, allowing I/O intensive trajectory analysis tasks to naturally exploit parallel filesystems.

VMD supports I/O-efficient MD simulation trajectory formats that have been specifically designed for so-called burst buffers and flash-based high-performance storage tiers, with emerging GPU-Direct Storage interfaces achieving I/O rates of up to 70 GB/s on a single GPU-dense DGX-2 compute node, and conventional parallel I/O rates approaching 1 TB/s. VMD startup scripts can be customized to run one or multiple MPI ranks per node, while avoiding undesirable GPU sharing conflicts, allowing compute, memory, and I/O resources to be apportioned among MPI ranks, and thereby best-utilized for the task at hand.

To facilitate large-scale simulation and analysis of biological systems containing millions of particles, VMD has been compiled on Frontera with MPI enabled. Frontera's capacity for massively parallel investigation of intact, authentic viral envelopes is demonstrated by application to envelope models at two levels of spatial resolution.

MODELING AUTHENTIC VIRAL ENVELOPES

CG Model of HIV-1 Envelope

Lipidomics profiling by mass spectrometry has established the lipid composition of the HIV-1 viral envelope.[7] Based on this experimental data, a CG model of an authentic HIV-1 envelope was constructed (see Figure 2A). The model exhibits a highly complex chemical makeup, with 24 different lipid species and asymmetry across the inner and outer leaflets of the membrane bilayer. The envelope is 150 nm in diameter and, with solvent, comprises 29 million CG beads, represented as MARTINI[8] particles. The CG model was equilibrated using GROMACS 2018.1,[9] and its dynamics were investigated over a production simulation time of over five microseconds. The MD simulation system, complete with solvent environment, is shown in Figure 2B.

Atomistic Model of HIV-1 Envelope

All-atom MD simulations are the most accurate classical mechanical approach that can be applied to study large-scale biological systems. Following simulation of the CG HIV-1 envelope model, which facilitated equilibration of lipid distributions over an extended simulation timescale, an atomistic model was constructed based on backmapping. The equilibrated CG model was mapped to an atomistic representation using the Backward tool,[5] which enables transformation of CG systems built with MARTINI. The approach utilizes a series of mapping files that contain structural information and geometric restrictions relevant to the modeling of each lipid species. Local geometries of atomistic lipids are reconstructed, taking into account stoichiometry and stereochemistry, to project the CG configuration into an atomistic configuration. An example of an atomistic representation of a lipid, along with its chemical structure, is given in Figure 2C. The final atomistic model of an authentic HIV-1 viral envelope comprises over 280 million atoms including solvent and ions. To complete the backmapping process, the system must be relaxed to a local energy minimum to resolve structural artifacts introduced by the CG to all-atom transformation.

VIRAL ENVELOPE ANALYSIS ON FRONTERA

Measuring Transverse Lipid Diffusion

Transverse diffusion of lipids occurs when they flip-flop between leaflets of the membrane

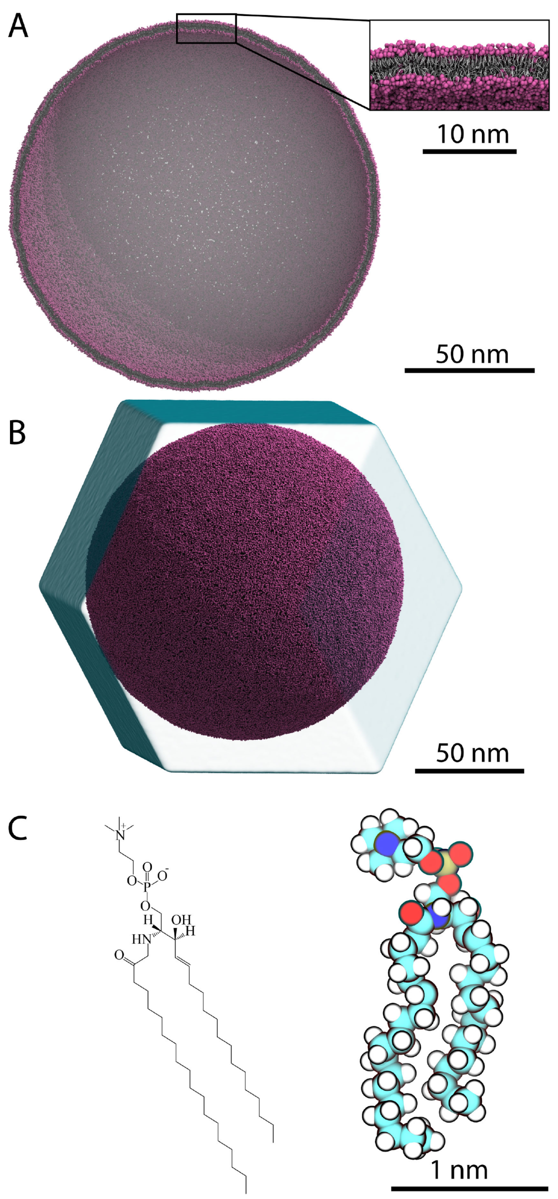


Figure 2. Authentic model of HIV-1 viral envelope **A** Cross-section of CG envelope model. Inset shows details of the inner (endoplasmic) and outer (exoplasmic) leaflets of the membrane bilayer. Envelope is 150 nm in diameter and comprises 29 million MARTINI particles. **B** CG envelope model suspended in physiological solvent environment, used for long-timescale MD simulations. **C** Chemical structure and atomistic representation of DPSM, one of 24 lipid species included in the HIV-1 envelope model. Each CG lipid in the envelope was backmapped to an atomistic version exhibiting full chemical detail. Visualizations of the virus structures were produced using VMD.[6]

bilayer. Cellular enzymes can catalyze this movement of lipids, or it can occur spontaneously over longer timescales. Computational approaches to characterize transverse diffusion in heterogeneous lipid mixtures have been described previously and indicate that flip-flop occurs at relatively slow rates. Current strategies to measure flip-flop events are based on tracking the translocation and reorientation of lipid headgroups (i.e., colored beads in Figure 2A).

A feature released in VMD 1.9.4 by the authors (*measure volinterior*)[10] facilitates tracking of headgroup dynamics by providing rapid classification of their locations in the inner versus outer leaflets of the bilayer. The method equips VMD with a mechanism to identify the inside versus outside of a biomolecular container. If the container surface is specified as the region of the bilayer composed of lipid tail groups (i.e., silver in Figure 2A), VMD can detect the presence of lipid headgroups in the endoplasmic versus exoplasmic leaflets. By tracking the number of headgroups that flip-flop from one leaflet to the other during intervals of simulation time, rates of transverse diffusion can be readily calculated.[10]

Frontera was leveraged to measure transverse lipid diffusion in the CG model of the authentic HIV-1 viral envelope using *measure volinterior* in VMD compiled with MPI support. The entire 5.2 μs of MD simulation, comprising 5200 frames with a storage footprint of 170 GB, was used for the calculation. To obtain high I/O performance, the trajectory files were set to a 1 MB stripe width, striping over 16 Lustre OSTs. The analysis was run on Frontera's primary compute partition, utilizing 56 Intel CLX cores per node.

Flip-flop events for the lipid N-stearoyl-D-erythro-sphingosylphosphorylcholine (DPSM, Figure 2C) were found to occur at a rate of $2.62 \times 10^{-4} \pm 1.49 \times 10^{-5}$ molecules per nanosecond from the outer to inner leaflet, and $2.33 \times 10^{-5} \pm 1.41 \times 10^{-6}$ molecules per nanosecond from the inner to outer leaflet (see Figure 3A). Because inward versus outward diffusion of this lipid species is not occurring at the same rate, the calculation reveals the process of lipid distributions being equilibrated across the asymmetric bilayer over the course of the simulation. The

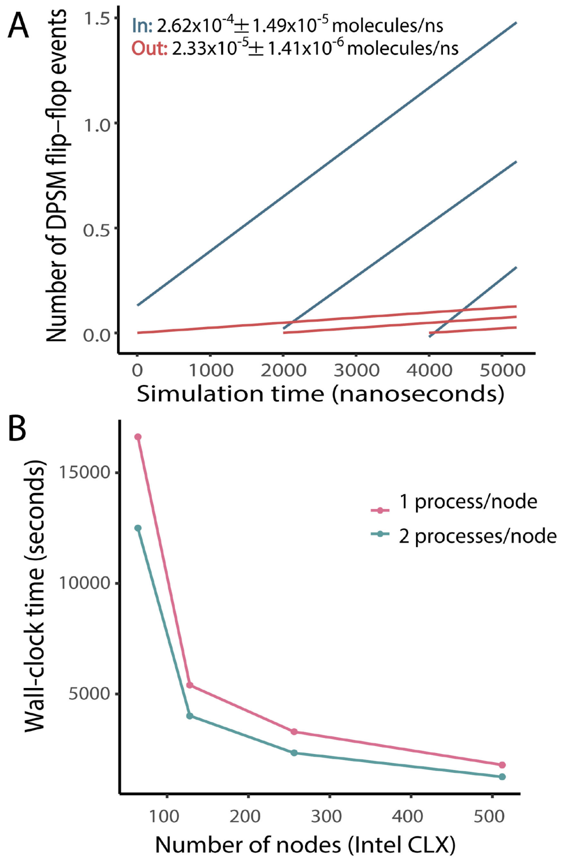


Figure 3. Parallel analysis of authentic viral envelopes. **A** Transverse diffusion analysis of the lipid DPSM in the CG model of the viral envelope. **B** Wall-clock time for calculation of transverse diffusion on Frontera.

Tcl framework for performing the calculation with VMD, the VMD startup script, and the SLURM job submission script used to run the calculation on Frontera are given in the Supplementary Information.

Figure 3B shows the wall-clock time for the transverse diffusion calculation with one VMD MPI rank per node and two VMD MPI ranks per node. By running multiple VMD MPI ranks per node, I/O operations are more effectively overlapped with computations, thereby achieving better overall I/O and compute throughout, both per node and in the aggregate, leading to an overall performance gain. It is worth noting, however, that finite per node memory, GPU, and interconnect resources ultimately restrict the benefit of using multiple MPI ranks per node to achieve better compute-I/O overlap. Ongoing VMD developments aim to improve compute-I/O overlap both for CPUs and for GPU-accelerated molecular modeling tasks through increased internal multithreading of trajectory I/O to allow greater asynchrony and decoupling of I/O from internal VMD computational kernels.

Iterative Relaxation of an Atomistic Envelope

The atomistic model of an authentic HIV-1 viral envelope presented here was derived via backmapping from a CG model. A major limitation of the backmapping approach involves structural artifacts introduced by the CG to all-atom transformation. Figure 4A shows an example of such structural artifacts in the lipid DPSM, including distorted bond lengths and angles. The current strategy for remedying these artifacts is to subject the backmapped model to molecular mechanics energy minimization, relaxing the geometry of each molecule, and the system as a whole, to a local energy minimum. Figure 4B shows an example of a postminimization structure in which the distortions have been successfully resolved.

Energy minimization is a common procedure applied to biomolecular systems to eliminate close contacts and nonideal geometries prior to the start of MD simulations. The energy of atomistic systems is described as an empirical potential energy function $V(r_i)$, where the terms depend on the positions r of i atoms. Here, energy minimization of the system is performed on Frontera using NAMD 2.14, an MD engine based on C++ and charmm++ that is amenable to large systems by virtue of being highly scalable on massive parallel computers.[11] Energy minimization is implemented in NAMD using the conjugate gradient algorithm for maximum performance. The atomistic model of an authentic HIV-1 viral envelope was subjected to energy minimization based on the CHARMM36 force field, with all the latest corrections in parameters for lipids and heterogenous protein-lipid systems.[12]

Figure 4C diagrams the complete procedure for constructing an atomistic envelope model from an initial CG model. Following backmapping of the CG representation to an all-atom representation, ions must be added to achieve electrostatic neutralization of the system, given the numerous charged lipid headgroups present in the envelope. Bulk ions are incorporated into

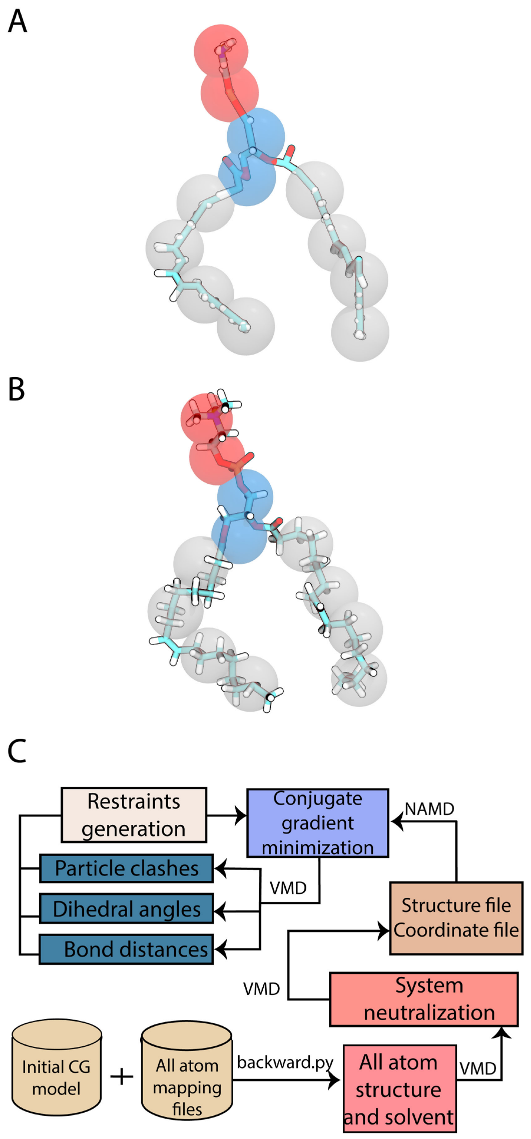


Figure 4. Resolving distortions in backmapped lipids. **A** Atomistic representation of DPSM with structural artifacts post-backmapping. CG representation of the lipid is superimposed to the all-atom structure. **B** Atomistic representation of DPSM post-minimization with structural artifacts resolved. CG representation of the lipid is superimposed to the relaxed all-atom structure. **C** Iterative relaxation process for large-scale authentic atomistic viral envelopes. Following backmapping from CG representations to all-atom, the system is subjected to charge neutralization, followed by generation of topology and coordinate files. Conjugate gradient minimization using NAMD, in combination with parallel analysis using VMD MPI, is employed iteratively to eliminate structural artifacts of the membrane and drive the system to a local minimum.

the explicit solvent, also backmapped from the CG model, to reproduce a realistic physiological environment. Coordinate and topology files are generated using the *js* plugin in VMD, utilizing a binary file format that overcomes the fixed-column limitation of *pdb* files. The minimization of multimillion atom systems can occasionally cause memory overflow or significant loss of performance in NAMD, due to the I/O, even on large-scale computational resources. In remedy, the memory-optimized version of NAMD implements parallel I/O and a compressed topology file, reducing the memory requirements for large biomolecular systems.

Following initial conjugate gradient minimization with NAMD, the model is assessed for major contributors to energetic instability, including structural clashes, close contacts, bond lengths, or angles significantly exceeding equilibrium values, and unfavorable dihedral configurations. Such assessments can be rapidly accomplished using VMD compiled with MPI support on Frontera, using existing VMD commands like *measure contacts*, *measure bond*, *measure angle*, and *measure dihed*. In a second iteration of minimization, regions of the system that are relatively stable are restrained in order to focus further minimization efforts on regions that remain the most energetically unfavorable. Useful NAMD commands for minimization of unstable systems include *minTinyStep* and *minBabyStep*, which control the magnitude of steps for the line minimizer algorithm for initial and further steps of minimization, respectively. This process is repeated until all geometric distortions are resolved, and the atomistic model has been driven to a local energy minimum, representative of the stable biological system at equilibrium.

CONCLUSIONS

Increased computational capability presents the opportunity for researchers to push the boundaries of scientific knowledge. Building, simulating, and analyzing models of intact, authentic viral envelopes, particularly at atomistic detail, is a relatively new research endeavor under active development. The leadership-class Frontera supercomputer provides a powerful resource to support the continuing growth of this field. The combination of Frontera's state-of-the-art, massively parallel architecture and high-performance parallel software applications, such as VMD, will enable researchers to make discoveries relevant to the treatment and prevention of disease at an unprecedented scale. All of the techniques discussed here are applicable to the study of other enveloped viruses, including SARS-CoV-2, which likewise comprises a complex lipidome. Notably, lipid compositions for viral envelopes can vary significantly depending on cell line and virus strain, and investigation of the effects of varying lipidomes on virus integrity and infectivity represents an interesting point of future research. Likewise, investigation of the interplay of viral envelopes with their embedded membrane proteins, as well as the viral-host membrane fusion process incorporating the complexity of both virus and cell lipidomes will become accessible as the field continues to mature. Owing to resources like Frontera, the computational biological sciences will play an increasingly important role in driving future innovations that improve human health and longevity.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation award MCB-2027096, funded in part by the Delaware Established Program to Stimulate Competitive Research (EPSCoR). This research is part of the Frontera computing project at the Texas Advanced Computing Center. Frontera is made possible by National Science Foundation award OAC-1818253. VMD development has been supported by National Institutes of Health Grant P41-GM104601. This research used resources provided by the Los Alamos National Laboratory Institutional Computing Program, which is supported by the U.S. Department of Energy National Nuclear Security Administration under Contract No. 89233218CNA000001.

REFERENCES

1. T. Reddy and M. S. Sansom, "Computational virology: From the inside out," *Biochimica et Biophys. Acta (BBA)-Biomembranes*, vol. 1858, no. 7, pp. 1610–1618, 2016.
2. J. A. Hadden and J. R. Perilla, "All-atom virus simulations," *Current Opinion Virology*, vol. 31, pp. 82–91, 2018.
3. J. K. Marzinek, R. G. Huber, and P. J. Bond, "Multiscale modelling and simulation of viruses," *Current Opinion Struct. Biol.*, vol. 61, pp. 146–152, 2020.
4. D. Stanzione, J. West, R. T. Evans, T. Minyard, O. Ghattas, and D. K. Panda, "Frontera: The evolution of leadership computing at the national science foundation," in *Proc. Pract. Exp. Adv. Res. Comput.*, 2020, pp. 106–111.
5. T. A. Wassenaar, K. Pluhackova, R. A. Bockmann, S. J. Marrink, and D. P. Tieleman, "Going backward: A flexible geometric approach to reverse transformation from coarse grained to atomistic models," *J. Chem. Theory Comput.*, vol. 10, no. 2, pp. 676–690, 2014.
6. W. Humphrey, A. Dalke, and K. Schulten, "VMD—Visual molecular dynamics," *J. Mol. Graph.*, vol. 14, no. 1, pp. 33–38, 1996.
7. M. Lorizate *et al.*, "Comparative lipidomics analysis of HIV-1 Particles and their producer cell membrane in different cell lines," *Cellular Microbiology*, vol. 15, no. 2, pp. 292–304, 2013.
8. S. J. Marrink, H. J. Risselada, S. Yefimov, D. P. Tieleman, and A. H. De Vries, "The MARTINI force field: Coarse grained model for biomolecular simulations," *J. Phys. Chem. B*, vol. 111, no. 27, pp. 7812–7824, 2007.
9. S. Pronk *et al.*, "GROMACS 4.5: A high-throughput and highly parallel open source molecular simulation toolkit," *Bioinformatics*, vol. 29, no. 7, pp. 845–854, 2013.
10. A. J. Bryer, J. A. Hadden-Perilla, J. E. Stone, and J. R. Perilla, "High-performance analysis of biomolecular containers to measure small-molecule transport, transbilayer lipid diffusion, and protein cavities," *J. Chem. Inf. Model.*, vol. 59, no. 10, pp. 4328–4338, 2019.
11. J. C. Phillips *et al.*, "Scalable molecular dynamics with NAMD," *J. Comput. Chem.*, vol. 26, no. 16, pp. 1781–1802, 2005.
12. J. B. Klauda *et al.*, "Update of the CHARMM all-atom additive force field for lipids: Validation on six lipid types," *J. Phys. Chem. B*, vol. 114, no. 23, pp. 7830–7843, 2010.

Fabio González-Arias is currently working in the Juan R. Perilla research group. He received the B.S. degree in chemistry and pharmaceutical chemistry with the Universidad Icesi, Cali, Colombia. He is currently working toward the Ph.D. degree with the University of Delaware, Newark, DE, USA. His research interests include large-scale molecular dynamics simulations, computational virology, and medicinal chemistry. Contact him at fabiogon@udel.edu

Tyler Reddy is currently a Staff Scientist with the CCS-7 (Applied Computer Science) group, Los Alamos National Laboratory, with a focus on open source (Python) software development, software build systems, computational geometry, and molecular dynamics simulations of enveloped viruses. He is a core developer of the SciPy and MDAnalysis libraries. He received the Ph.D. degree in biochemistry and molecular biology. Contact him at treddy@lanl.gov.

Jodi A. Hadden-Perilla is currently an Assistant Professor with the Department of Chemistry and Biochemistry, University of Delaware, Newark, DE, USA. She received the Ph.D. degree in computational chemistry from the University of Georgia, Athens, GA, USA, in 2014, and carried out Postdoctoral Training with the Beckman Institute, University of Illinois at Urbana-Champaign. Her research group uses molecular dynamics simulations to study biological machines, including viruses, and molecular motors. He is the corresponding author of this article. Contact her at jhadden@udel.edu.

John E. Stone is the lead developer of VMD, a high performance tool for preparation, analysis, and visualization of biomolecular simulations used by over 100 000 researchers all over the world. His research interests include molecular visualization, GPU computing, parallel computing, ray tracing, haptics, virtual environments and immersive visualization. He is an NVIDIA CUDA Fellow. He is a member of the Khronos working group for the ANARI analytic rendering standard for high-fidelity scientific and technical visualization, and is a member of the advisory panel for the Vulkan graphics API. He is a member of ACM, ACM Siggraph, and IEEE. Contact him at johns@ks.uiuc.edu.

Juan R. Perilla is currently an Assistant Professor with the Department of Chemistry and Biochemistry, University of Delaware, Newark, DE, USA. His research group focuses on developing and applying high performance computational tools to determine the molecular details of life and disease. He received the Ph.D. degree in biophysics from the Johns Hopkins University, Baltimore, MD, USA. He is a member of the Biophysical Society, American Physical Society, and American Chemical Society. He is the corresponding author of this article. Contact him at jperilla@udel.edu.

IEEE SECURITY & PRIVACY
SOFTWARE EVERYWHERE
IEEE Symposium on Security and Privacy
ECONOMICS OF CYBERSECURITY PART 2
S&P

Revealing the Mechanism of SARS-CoV-2 Spike Protein Binding With ACE2

Yixin Xie, Dan Du, Chitra B. Karki, Wenhan Guo, Alan E Lopez-Hernandez, Shengjie Sun, Brenda Y Juarez, Haotian Li, Jun Wang, and Lin Li
University of Texas at El Paso

***Abstract*—A large population in the world has been infected by COVID-19. Understanding the mechanisms of Severe Acute Respiratory Syndrome CoronaVirus 2 (SARS-CoV-2) is important for the management and treatment of COVID-19. When it comes to the infection process, one of the most important proteins in SARS-CoV-2 is the spike (S) protein, which is able to bind to human Angiotensin-Converting Enzyme 2 (ACE2) and initializes the entry of the host cell. In this study, we implemented multiscale computational approaches to study the electrostatic features of the interfaces of the SARS-CoV-2 S protein receptor binding domain and ACE2. The simulations and analyses were performed on high-performance computing resources in the Texas Advanced Computing Center. Our study identified key residues on SARS-CoV-2, which can be used as targets for future drug design. The results shed light on future drug design and therapeutic targets for COVID-19.**

The number of confirmed cases of Coronavirus Disease 2019 (COVID-19) is increasing dramatically[1] due to the fast spread of SARS-CoV-2. The large coronavirus family includes hundreds of viruses that usually do not pose a threat to human health. SARS-CoV-2 is the seventh member of those coronaviruses that infect the human body. Of these, four (HCoV-229E, HCoV-OC43, HCoV-NL63, HKU1)[2] cause mild to moderate symptoms, while the other three can cause serious, even fatal diseases. SARS coronavirus (SARS-CoV) broke out in 2002 and caused Severe Acute Respiratory Syndrome (SARS). MERS coronavirus (MERS-CoV)

Digital Object Identifier 10.1109/MCSE.2020.3015511
Date of publication 11 August 2020; date of current version 9 October 2020.

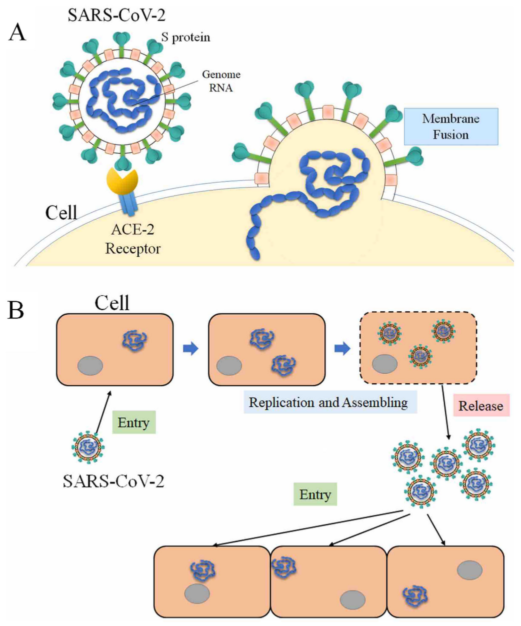


Figure 1. Process of SARS-CoV-2 infecting host cells. (A) SARS-CoV-2 S protein binds to ACE2 of host cell, and enters the host cell after binding. (B) SARS-CoV-2 utilizes the host cell as a factory to reproduce more SARS-CoV-2 and release them to infect more cells.

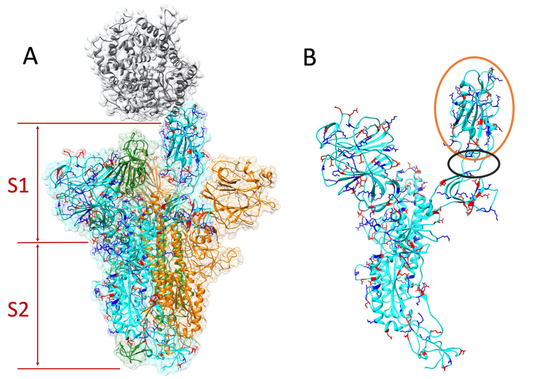


Figure 2. Structure of S protein homotrimer and ACE2. Positive and negative residues on a monomer of S protein are colored in blue and red, respectively. (A) ACE2 is colored in gray. The S protein trimer contains three identical monomers, colored in cyan, green, and orange. One monomer (cyan)'s RBD flips out to bind with the ACE2. The S protein is composed by S1 and S2 subunits. (B) Single S protein monomer structure. The orange circle shows the RBD and the black circle marks the flexible hinge connecting RBD and other part of the S protein.

found in 2012 caused Middle East Respiratory Syndrome (MERS), which continues to cause sporadic and localized outbreaks. The animal-to-human and human-to-human transmissions of these two fatal viruses caused the outbreak of the 2003 SARS pandemic and the following 2012 MERS pandemic, which resulted in over 8000 and 2400 reported cases, and killed 774 and 858 people, respectively. SARS-CoV-2 is the third known death-leading coronavirus that appeared in 2019, has caused a global pandemic, and compared to SARS and MERS, COVID-19 spreads much faster and infects a larger population globally. Many studies have been made on SARS-CoV-2 to develop COVID-19 diagnostics, therapeutics, and vaccines. Meanwhile, a great number of fundamental studies have also been focused on revealing the mechanisms of SARS-CoV-2 and other coronaviruses to deeply understand them and assist the treatment and drug design for COVID-19.

Coronaviridae is a family of enveloped, positive-sense, single-stranded RNA viruses that infects a wide range of hosts, including amphibians, birds, and mammals.[3] The spherical viruses are typically decorated with spike (S) proteins, making the whole virus the shape of solar corona. The structure of a coronavirus mainly contains several types of proteins, including: the nucleocapsid (N) proteins which are inside the envelope of the coronavirus, the membrane (M) proteins and the envelope (E) proteins which form the virus envelope, and the spike (S) proteins which are trimmers on the envelope of the virus. The spike (S) proteins initialize an infection by binding to the human Angiotensin-Converting Enzyme 2 (ACE2). Then, SARS-CoV-2 enters the host cell and reproduces more viruses, which are released later to infect more cells (see Figure 1). Inhibiting the interactions between S proteins and ACE2 can block the infections, which is a direction for therapeutic drug design. Therefore, many of studies have been focused on the interactions of coronaviruses' S proteins and human ACE2.

Computational approaches are successfully used to study protein–protein interactions.[4–6] Such approaches can also be used to study the protein–protein interactions of viruses. Computations of viruses are challenging due to the complexity of the viruses. Two categories of computational approaches have been implemented to study the viruses: atomic simulations, which study a whole virus or part of a virus at the atomic level, and coarse-grained simulations, which treat a virus at the amino acid level. Some research efforts are focused on determining the structures of viruses, others reveal fundamental mechanisms for viruses, such as viral capsid assembly[7] and viral capsid electrostatic features.[8] Due to the fast growth of supercomputer performance and rapid developments of computational algorithms, computational approaches play significant roles in virus studies.

This article studies the fundamental mechanisms of SARS-CoV-2 S protein binding to ACE2. Based on the structure of the SARS-CoV-2 S protein and ACE2 (see Figure 2), several computational approaches were utilized to perform calculations and simulations. These approaches can be widely utilized to study more viruses. The electrostatic features were analyzed based on the distribution of the charged residues. The hydrogen bonds and salt bridges are studied after molecular dynamic (MD) simulations. Key residues, which significantly contribute to binding energy, are also identified in this work. Previous studies demonstrated that the interactions between ACE2 and SARS-CoV-2 is more temperature-sensitive than ACE2 and SARS-CoV. The binding affinity of ACE2 and SARS-CoV-2 was also calculated.[9] In this article, more detailed binding mechanisms, such as the key residues, salt bridges, and hydrogen bonds are revealed. Such details provide more specific information for future drug design. The findings in this article will pave the way for other research related to drug design and treatments for COVID-19 and other coronavirus-caused diseases.

This article studies the fundamental mechanisms of SARS-CoV-2 S protein binding to ACE2.

METHODS

Structure Preparation

The genetic sequence of SARS-CoV-2 was obtained from Genebank,[10] which was from the early patients in December 2019. The SARS-CoV-2 with the ACE2 complex structure was modeled by the Swiss-Model web server (https://swissmodel.expasy.org/) based on the template of the complex structures of SARS-CoV and ACE2 (pdb ID: 6ACG, https://www.rcsb.org/structure/6ACG). To study their electrostatic interactions, we mainly focus on the receptor binding domain (RBD) of the S protein and the binding domain of ACE2.

Electrostatic Calculations Using DelPhi

In order to study the electrostatic features, DelPhi[11] was utilized to calculate the electrostatic potential for the S protein RBD and ACE2 binding domain. In the framework of continuum electrostatics, DelPhi calculates the electrostatic potential ϕ (in systems comprised of biological macromolecules and water in the presence of mobile ions) by solving the Poisson–Boltzmann equation (PBE)

$$\nabla \cdot [\epsilon(\mathrm{r})\nabla\phi(\mathrm{r})] = -4\pi\rho(\mathrm{r}) + \epsilon(\mathrm{r})\kappa^2(\mathrm{r})\sinh(\phi(\mathrm{r})/\mathrm{k_B T}) \quad (1)$$

where $\phi(\mathrm{r})$ is the electrostatic potential, $\epsilon(\mathrm{r})$ is the dielectric distribution, $\rho(\mathrm{r})$ is the charge density based on the atomic structures, κ is the Debye–Huckel parameter, $\mathrm{k_B}$ is the Boltzmann constant, and T is the temperature. Due to the irregular shape of macromolecules, DelPhi uses a finite difference method to solve the PBE.

The electrostatic potential of the SARS-CoV-2 S protein with ACE2 was calculated by DelPhi. The calculated electrostatic potential on the surface was visualized with Chimera (see Figure 3). In order to visualize electric field lines between SARS-CoV-2 and ACE2, the visual molecular dynamics (VMD: https://www.ks.uiuc.edu/Research/vmd/) software package (see Figure 3) was implemented based on the electrostatic potential map from DelPhi calculations. The color scale range was set from –1.0 to 1.0 kT/Å.

MD Simulations SARS-CoV-2 RBDs

To simulate the dynamic interactions between S proteins and ACE2 protein, 20-ns MD simulations were carried out using NAMD on GPUs using Lonestar5 at the Texas Advanced Computing Center (TACC https://www.tacc.utexas.edu/). A 2000-step minimization was performed for each simulation, followed by a 51 000-step simulation to relax the complex structures to avoid the clashes at the binding interface. During the MD simulations, the

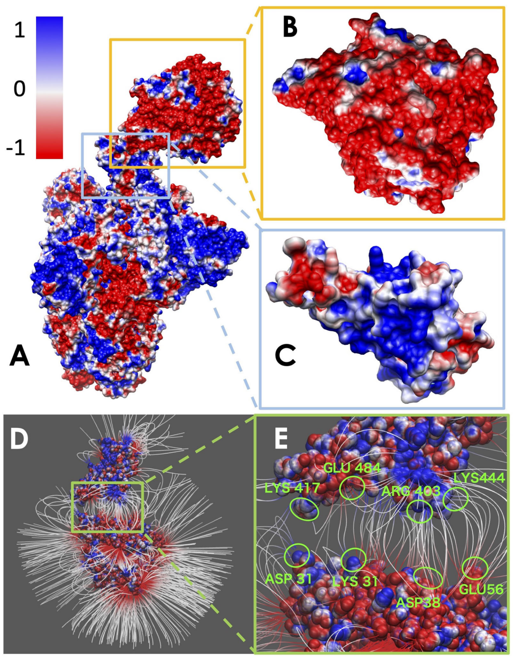


Figure 3. Electrostatic surface and electric field lines of SARS-CoV-2 S protein and ACE2 binding domain. (A) Electrostatic surface of S protein and ACE2. (B) Binding interface of ACE2. (C) Binding interface of S protein RBD. (D) Electric field lines of S protein RBD and ACE2 binding domain. (E) A closeup of electric field lines at the binding interfaces. The residues forming salt bridges are highlighted.

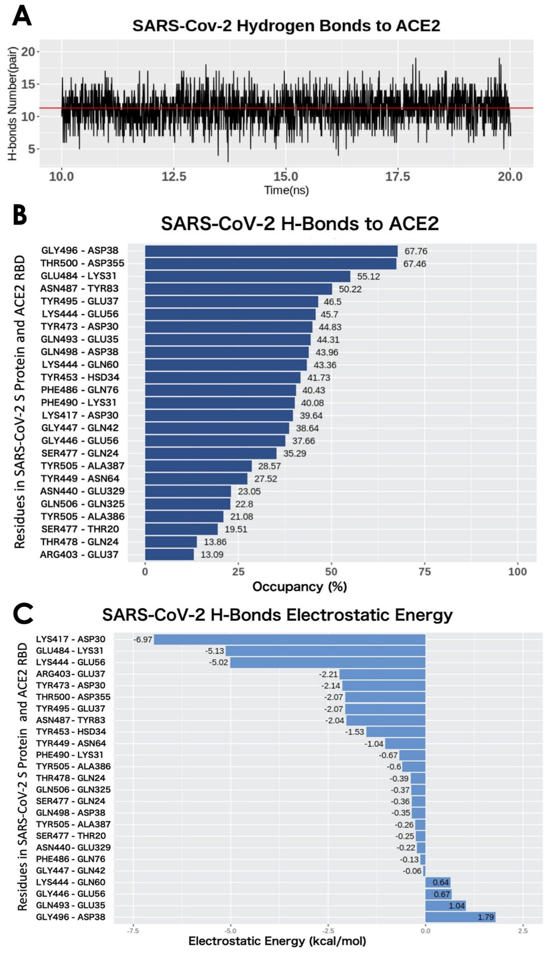


Figure 4. H-bonds between SARS-CoV-2 RBD and ACE2 binding domain. (A) Numbers of H-bonds at different time stamps from 10 to 20 ns, the average value is marked as red line with the value of 11.30. (B) Occupancy of H-bonds ordered by decreasing values of occupancy, and the residues are labeled in *y*-axis with the SARS-Cov-2 in the left and ACE2 in the right. (C) List of hydrogen bonds that are ranked by their electrostatic energy values.

temperature was set to be 300K, and the pressure was set as standard using the Langevin dynamics. For PME, which is set for full-system periodic electrostatics, the grid size is (86, 88, 132) as *x*, *y*, and *z* values, respectively. After the first 0.05 ns of the simulation, atoms that were not involved in binding domains were constrained within a margin of 5.0 Å of their natural movement maximum length value, since the binding forces were strong enough for proteins to bond completely after 51 000 steps, which is not suitable for finding salt bridges. In order to get a more accurate result of the simulation, data of the last 10 ns of simulations were used for data analysis.

To analyze the binding interaction between S protein and ACE2 by finding the key residues, the salt bridges that were formed within the distance of 4 Å were extracted from the last 2000 frames of simulations, and for hydrogen bonds, the cutoff was 3.2 Å. The several top-strongest salt bridges in each binding domain were determined by calculating their formation frequency (occupancy in Figure 4) during MD simulation. The simulation is shown in Movie 1.

Binding Energy Calculations Using MM/PBSA

For the binding energy calculations of the S protein and ACE2, the widely used MM/PBSA method was involved in calculating the average binding energy from the last 2000 frames, which were the frames of the last 10 ns trajectory of the MD simulation (see Tables 1 and 2).

Table 1. Binding Energy of SARS-CoV-2 S Protein RBDs Binding With ACE2 Binding Domain. SD Stands for Standard Deviation.

Energy terms	Magnitude (kcal/mol)	
	Mean	SD
Polar solvation energy	630.18	11.7
Nonpolar solvation energy	–18.41	0.41
Van der Waals energy	–128.53	8.43
Coulombic energy	–656.69	16.04
Electrostatic energy	–26.51	10.48

In the MM/PBSA calculation method, ΔG_{bind} is calculated as

$$\Delta G_{\text{bind}} = G_{\text{complex}} - G_{\text{virus-bd}} - G_{\text{ACE2}} \tag{2}$$

where G_{complex} is the total energy of complex of S protein and ACE2 pair; $G_{\text{virus-bd}}$ and G_{ACE2} are the total energies of the two individual proteins, and virus-bd stands for the binding domain of SARS-CoV-2 to ACE2. The total energy is calculated as

$$G_{\text{total}} = G_{\text{coul}} + G_{\text{polar}} + G_{\text{vdw}} + G_{\text{nonpolar}} \tag{3}$$

where G_{coul} is the Coulombic energy, G_{polar} is the polar part of the solvation energy, G_{vdw} is the Van der Waals energy, and G_{nonpolar} is the nonpolar part of the solvation energy. The Coulombic and polar electrostatic energies were calculated by DelPhi. The Van der Waals binding energy was calculated using NAMD. The nonpolar term of solvation energy was calculated via the solvent accessible surface area method

$$G_{\text{nonpolar}} = \gamma SA + b \tag{4}$$

where $\gamma = 0.0054\ \text{kcal} \cdot \text{mol}^{-1} \cdot \text{A}^{-2}$, $b = 0.92\ \text{kcal} \cdot \text{mol}^{-1} \cdot \text{A}^{-2}$,[12] and SA denotes the solvent accessible surface area, which is calculated using the Naccess2.1.1 program (http://www.bioinf.manchester.ac.uk/naccess/).

RESULTS AND DISCUSSIONS

The structure of SARS-CoV-2 S protein binding with human ACE2 is shown in Figure 2. The S protein is a homotrimer that contains three identical monomers, as shown in cyan, orange, and green. ACE2 is shown in gray. The main

Table 2. Binding Energy of Individual Hydrogen Bonds Between SARS-CoV-2 S Protein RBD and ACE2 Binding Domain, With the Decreasing Order of the Electrostatic Energy Mean Values.

H-Bonds	Electrostatic energy (kcal/mol)		Hydrogen bonds Occupancy
	Mean	**SD	(%)
*LYS417-ASP30	–6.97	1.57	39.64
*GLU484-LYS31	–5.13	1.53	55.12
*LYS444-GLU56	–5.02	1.67	45.7
*ARG403-GLU37	–2.21	2.4	13.09
TYR473-ASP30	–2.14	2.28	44.83
THR500-ASP355	–2.07	1.52	67.46
TYR495-GLU37	–2.07	1.65	46.5
ASN487-TYR83	–2.04	1.57	50.22
TYR453-HSD34	–1.53	1.29	41.73
TYR449-ASN64	–1.04	1.38	27.52
PHE490-LYS31	–0.67	1.12	40.08
TYR505-ALA386	–0.6	1.45	21.08
THR478-GLN24	–0.39	1.46	13.86
GLN506-GLN325	–0.37	0.87	22.8
SER477-GLN24	–0.36	2.13	35.29
GLN498-ASP38	–0.35	0.94	43.96
TYR505-ALA387	–0.26	1.73	28.57
SER477-THR20	-0.25	1.12	19.51
ASN440-GLU329	–0.22	0.92	23.05
PHE486-GLN76	–0.13	0.75	40.43
GLY447-GLN42	–0.06	0.72	38.64
LYS444-GLN60	0.64	1.58	43.36
GLY446-GLU56	0.67	0.75	37.66
GLN493-GLU35	1.04	1.06	44.31
GLY496-ASP38	1.79	1.07	67.76

**Salt bridges at the binding interface. **SD stands for Standard Deviation. For each pair of the hydrogen bond, residue ID on the left side is from SARS-CoV-2 S protein RBD and the right side is from ACE2 binding domain.*

function of the S1 subunit is to bind its RBD with ACE2, while the S2 subunit mainly focuses on the cell membranes' fusion between the virus and host cell. The structure shows that the RBD from the S protein monomer (cyan) flips out and binds with the ACE2, while the other two monomers' RBDs are hidden in the S1 subunit. This configuration change of RBD is essential for the S protein binding to ACE2. The hinge structure responsible for the configuration change is shown in the black circle in Figure 2(B).

The electrostatic features are important for the protein–protein binding process. Therefore, we indicated the positive and negative residues in blue and red, respectively, on one of the S protein monomers [see Figure 2(A) and (B)]. Most of the charged residues are distributed on the surface of the S protein. At the binding interface of the RBD, the dominant charge is positive, which provides attractive force to ACE2, as the net charge of the ACE2 binding interface is negative. Details are shown in electrostatic surfaces and electric field lines in the following sections.

Electrostatic Surfaces

Electrostatic features are important for protein–protein interactions and drug design. We calculated the electrostatic potential on the surface of the SARS-CoV-2 S protein and ACE2. The binding interface of ACE2 shows significant negative electrostatic potential, while the binding interface of SARS-CoV-2 S protein RBD shows dominantly positive potential (see Figure 3). Therefore, the S protein RBD is attracted by the ACE2 due to their opposite net charges at their binding interfaces. Such electrostatic binding force is common in other strong protein–protein interactions, which provides long-range interactions.

To further illustrate the electrostatic interactions, electric field lines between the SARS-CoV-2 S protein and ACE2 are calculated and shown in Figures 3(D) and (E). Dense electric field lines indicate strong electrostatic forces. From the electric field line distribution, it is clear that there are four groups of dense field lines between the S protein RBD and ACE2. The sources of the four groups of field lines are all charged residues, which are labeled in Figure 3(E). Those residues are further proved to be the salt-bridge residues at the binding interfaces of S protein RBD and ACE2, as shown in Figure 3(E).

Binding Energy

Since SARS-Cov-2 attracts hACE2 to bind, the binding energy is one of the essential terms that can show how much contribution each different residue makes in the binding process. The binding energy of the SARS-CoV-2 S protein RBD and ACE2 binding domain are calculated using the MM/PBSA method. The results of the binding energy are shown in Table 1. More detailed calculations and analyses of salt bridges and hydrogen bonds are shown in Table 2. In Table 1, the nonpolar solvation energy is the energy of excluding the water molecules when the protein complex immerses in water. The Van der Waals energy decreases significantly when the S protein RBD is away from ACE2. The nonpolar solvation energy and Van der Waals binding energy are nonspecific to amino acids. These two energy terms mainly depend on the binding surface area. Normally, a larger binding surface results in greater nonpolar solvation energy and Van der Waals binding energy values. In this article, we mainly focus on the long-range electrostatic binding energy, which is the sum of the Coulombic energy and polar solvation energy. These two energy terms strongly depend on the distribution of amino acids, especially the charged amino acids. For the SARS-CoV-2 S protein RBD and ACE2, the electrostatic binding energy is –26.51 kcal/mol. To further examine which residues significantly contribute to the electrostatic binding energy, we performed the analyses on hydrogen bonds and salt bridges.

Since SARS-Cov-2 attracts hACE2 to bind, the binding energy is one of the essential terms that can show how much contribution each different residue makes in the binding process.

Hydrogen Bonds

The drug targets usually are the residues that form hydrogen bonds or salt bridges. Therefore, studying hydrogen bonds and salt bridges at the binding interfaces is crucial for COVID19. Hydrogen bonds and salt bridges at the interfaces of S protein and ACE2 are calculated based on the MD simulations and are analyzed using the VMD H-bond tool. The average number of H-bonds is 11.30, shown as the red lines in Figure 4(A). From Figure 4(B), the H-bonds with over 10% occupancy are presented with a decreasing order of occupancy. Among the 25 pairs of hydrogen bonds, four salt bridges are included and marked with asterisks. The occupancy of each hydrogen bond pair is shown in Table 2 and visualized in Figure 4(B). The highest occupancy of the salt bridge is 67.76%, from the S protein GLY469 and ACE2 ASP38. Another hydrogen bonds' list is shown in Figure 4(C), which is ranked by the electrostatic binding energy contributed by each pair of hydrogen bond. The top four pairs are LYS417-ASP30, GLU484-LYS31, LYS444-GLU56, ARG403-GLU37 (for each pair, the residue id on the left side is from the SARS-CoV-2 S protein and that on the right side is from ACE2). By default, the VMD

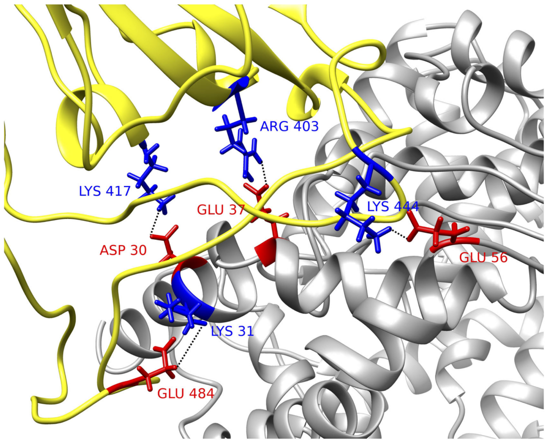


Figure 5. Structural demonstration of key residues in salt bridges. SARS-CoV-2 S protein RBD (yellow) with human ACE2 binding domain (gray). Key residues are marked with its amino acid types, sequence numbers, and charges, where blue stands for positively charged amino acid and red stands for the negative ones. Four pairs of salt bridges are found by VMD. All the structures in this figure are rendered by Chimera.

Salt Bridge tool identifies a salt-bridge when the distance between a positive residue and a negative residue is less than 4Å, no matter if these two residues form hydrogen bond or not. It is notable that the top four hydrogen bonds are also salt bridges. We therefore can conclude that the S protein is attracted to ACE2, and there are a total of 25 residue pairs between the interfaces that contribute significantly to the electrostatic binding energy. The top four residue pairs are all salt bridges. The average distance between each pair of residues in these top salt bridges is calculated: LYS417-ASP30, 2.5Å; GLU484-LYS31, 4.4Å; LYS444-GLU56, 2.8Å; ARG403-GLU37, 2.6 Å. The sum of the electrostatic binding energies of the four salt bridges is –19.33 kcal/mol, which takes 73% of the total electrostatic binding energy. Thus, the salt-bridge involved residues are the most important residues that can be considered as future drug targets. The structures of the four salt bridges at the interfaces are illustrated in Figure 5.

CONCLUSION

COVID-19 infects a large population in the world. This fatal disease is caused by SARS-CoV-2, a novel coronavirus. Understanding the mechanisms of SARS-CoV-2 is critically important, not only for ongoing COVID-19 but also for the upcoming challenges from other disease-causing coronaviruses in the future. A deep understanding of the mechanisms of this virus is critical to developing treatments and vaccines for SARS-CoV-2, and to be better prepared for potential future novel viruses.

Among the types of proteins in coronaviruses, S protein plays a significant role of attaching the virus to the host cell. Blocking the interactions between S protein and human ACE2 may result in effective therapeutic targets for COVID-19. This study implemented several computational approaches to study the fundamental mechanisms of COVID-19 S protein binding with human ACE2. The electrostatic features of COVID-19 S protein and ACE2 are analyzed according to the charge distribution on their structures. Hydrogen bonds and salt bridges on the interfaces are studied based on MD simulations. Residues involved in salt bridges are identified as key residues that stabilize the interactions of S protein and ACE2 complex structure. Blocking these key residues may inhibits the function of S protein, thus preventing the infection of SARS-CoV-2. This study provides potential targets for the drug design of COVID-19. The methods implemented in this research can be widely used to study other viruses.

ACKNOWLEDGMENTS

The calculations and analyses were performed at the Texas Advanced Computing Center. This work was supported in part by the National Institutes of Health under Grant SC1GM132043-01; in part by the National Institutes on Minority Health and Health Disparities, a component of the NIH under Grant 5U54MD007592.

REFERENCES

1. World Health Organization, *Coronavirus Disease 2019 (COVID-19): Situation Report, 85*, 2020.
2. A. R. Fehr and S. Perlman, “Coronaviruses: An overview of their replication and pathogenesis,” in *Coronaviruses*. Berlin, Germany: Springer, 2015, pp. 1–23.

3. G. Li *et al.*, “Coronavirus infections and immune responses,” *J. Med. Virol.*, vol. 92, no. 4, pp. 424–432, 2020.
4. L. Li, L. Wang, and E. Alexov, “On the energy components governing molecular recognition in the framework of continuum approaches,” *Frontiers Mol. Biosci.*, vol. 2, 2015, Art. no. 5.
5. L. Li, J. Alper, and E. Alexov, “Cytoplasmic dynein binding, run length, and velocity are guided by long-range electrostatic interactions,” *Sci. Rep.*, vol. 6, 2016, Art. no. 31523.
6. L. Li, J. Alper, and E. Alexov, “Multiscale method for modeling binding phenomena involving large objects: Application to kinesin motor domains motion along microtubules,” *Scientific Rep.*, vol. 6, 2016, Art. no. 23249.
7. Y. Xian *et al.*, “The roles of electrostatic interactions in capsid assembly mechanisms of giant viruses,” *Int. J. Mol. Sci.*, vol. 20, no. 8, 2019, Art. no. 1876.
8. C. Li *et al.*, “Highly efficient and exact method for parallelization of grid-based algorithms and its implementation in DelPhi,” *J. Comput. Chem.*, vol. 33, no. 24, pp. 1960–1966, 2012.
9. J. He *et al.*, “Molecular mechanism of evolution and human infection with sars-cov-2,” *Viruses*, vol. 12, no. 4, 2020, Art. no. 428.
10. F. Wu *et al.*, “A new coronavirus associated with human respiratory disease in China,” *Nature*, vol. 579, no. 7798, pp. 265–269, 2020.
11. L. Li *et al.*, “DelPhi: A comprehensive suite for DelPhi software and associated resources,” *BMC Biophysics*, vol. 5, no. 1, 2012, Art. no. 9.
12. D. Sitkoff, K.A. Sharp, and B. Honig, “Accurate calculation of hydration free energies using macroscopic solvent models,” *J. Phys. Chem.*, vol. 98, no. 7, pp. 1978–1988, 1994.

Yixin Xie received the B.S. degree in computer science from Southern Polytechnic State University, Marietta, GA, USA, in 2014 and the M.A. degree in media arts from the University of Technology, Sydney, Ultimo NSW, Australia, in 2016. She is currently working toward the Ph.D. degree in computational science with the University of Texas at El Paso, El Paso, TX, USA. Her research is focused on utilizing comprehensive computational approaches to study molecular motors and disease-causing viruses including SARS-CoV-2. Contact her at yxie4@miners.utep.edu.

Dan Du received the B.S. degree in electronic information from Central China Normal University, Wuhan, China, in 2007, the M.S. degree in physics from Huazhong University of Science and Technology, Wuhan,China, in 2011, and she received the M.S. degree in computer engineering from Clemson University, Clemson, SC, USA, in 2016. She is currently working toward the Ph.D. degree in computational science with the University of Texas at El Paso, El Paso, TX, USA. Her research interests include DNA/RNA secondary structure predictions and mechanisms of viral infections such as SARS-CoV-2. Contact her at ddu2@miners.utep.edu.

Chitra B. Karki received the B.S. degree in physics from Prithivi Narayan Campus, Tribhuvan University, Nepal, in 2011, the M.S. degree in physics from the University Campus, Tribhuvan University, Pokhara, Nepal, in 2013, and the M.S. degree in physics in 2019 from the University of Texas at El Paso, where he is currently working toward the graduate degree with the Computational Science Department. His research interest is in understanding different disease related proteins using various computational approaches. Contact him at cbkarki@miners.utep.edu.

Wenhan Guo received the M.S. degree in public health from Zhengzhou University, Zhengzhou, China, in 2018, and is currently working toward the master’s degree with the Computational Science Department, University of Texas at El Paso, El Paso, TX, USA. Her research is mainly focused on the study of the structure of virus and scientific software development. Contact her at wguo1@miners.utep.edu.

Alan E Lopez-Hernandez received the B.S. degree in physics from University of Texas at El Paso in 2020, and is currently working toward the master’s degree with the Computational Science Department, University of Texas at El Paso, El Paso, TX, USA. His research is mainly focused on using computational approaches to study virus biophysical properties. Contact him at aelopezhern@miners.utep.edu.

Shengjie Sun received the B.S. degree in physics from Central South University, Changsha, China, in 2016, and is currently working toward the Ph.D. degree with the Computational Science Department, University of Texas at El Paso, El Paso, TX, USA. His research interests are in the study of the interaction between myosin and actin and scientific software development. Contact him at ssun1@miners.utep.edu.

Brenda Y Juarez is currently working toward the bachelor's degree in physics with a pre-med concentration with the University of Texas at El Paso, El Paso, TX, USA. She wants to further her studies and pursue the M.D./Ph.D. degrees with a Ph.D. concentration on biophysics. Contact her at byjuarez@miners.utep.edu.

Haotian Li's main research includes biomolecular structure prediction and related method development. He received the B.S. degree in physics and the Ph.D. degrees in physics from the Huazhong University of Science and Technology, Wuhan, China, in 2011 and 2019, respectively. Contact her at 767265653@qq.com.

Jun Wang received the B.S. degree in physics the Ph.D. degree in physics major from the Huazhong University of Science and Technology, Wuhan, China, in 2009 and 2017, respectively. His main research focuses on developing programs to solve computational biology problems. Contact him at junwangwx@gmail.com.

Lin Li is currently an Assistant Professor with the Physics Department, University of Texas at El Paso, El Paso, TX, USA. His research is mainly focused on developing and applying novel computational approaches to study health related biology problems. He received the B.S. and Ph.D. degrees in physics from the Huazhong University of Science and Technology, Wuhan, China, in 2005 and 2011, respectively. He worked as a Postdoctoral Researcher, Research Associate, and then Research Assistant Professor with Clemson University from 2011 to 2017. Since 2017, he has been working at the University of Texas at El Paso as an Assistant Professor. He has authored or coauthored more than 40 peer reviewed scientific papers in the area of computational biophysics. His research bridges physics, computing, and biology. He is the corresponding author of this article. Contact him at lli5@utep.edu.

Theme Article: Computational Science in the Battle Against COVID-19

Biomolecular Simulations in the Time of COVID-19, and After

Rommie E. Amaro
UC San Diego

Adrian J. Mulholland
University of Bristol

***Abstract*—COVID-19 has changed life for people worldwide. Despite lockdowns globally, computational research has pressed on, working remotely and collaborating virtually on research questions in COVID-19 and the virus it is caused by, SARS-CoV-2. Molecular simulations can help to characterize the function of viral and host proteins and have the potential to contribute to the search for vaccines and treatments. Changes in the *modus operandi* of research groups include broader adoption of the use of preprint servers, earlier and more open sharing of methods, models, and data, the use of social media to rapidly disseminate information, online seminars, and cloud-based virtual collaboration. Research funders and computing providers worldwide recognized the need to provide rapid and significant access to computational architectures. In this article, we discuss how the interplay of all of these factors is influencing the impact—both potential and realized—of biomolecular simulations in the fight against SARS-CoV-2.**

In January 2020, few people could have envisioned how drastically the world as we knew it would be upended, and how quickly, due to COVID-19. Unusual cases of viral pneumonia were first identified in Wuhan, China, at the end of 2019. The cause was determined to be a novel coronavirus (SARS-CoV-2). The first cases subsequently appeared in the U.S. in late January 2020. By February, cases in Europe were spreading concerningly, particularly in the Lombardy region in Italy. Lockdowns were imposed in many countries in response. By mid-March, California went into lockdown, and the U.K. followed suit a short time later. Halfway through 2020, lockdowns have happened all across the

Digital Object Identifier 10.1109/MCSE.2020.3024155
Date of publication 15 September 2020; date of current version 9 October 2020.

world. By July 1, the U.S. hit a milestone of 50 000 new confirmed COVID-19 cases in a single day. In the first half of 2020 alone, over half a million people have died from this pandemic disease and it continues to spread globally.

Scientists the world over are working to meet this challenge, in the face of added obstacles posed by lockdowns. Research funders, companies, and other organizations are making impressive efforts and commitments to share and analyze scientific and biomedical data, which have emerged rapidly in the face of the pandemic.* Computational researchers have to some extent been less affected by lockdowns than their experimental counterparts. Computational chemists and biologists can run jobs remotely on high-performance computers (HPCs), or in the cloud, regardless of whether the scientist is in their office at a university/institute or from their kitchen table in their homes. Thus, many computational scientists in the fields of biology, chemistry, medicine, and allied fields realized the chance to make potentially significant near- and long-term impact. Numerous groups pivoted their efforts to address the COVID-19 challenge, seizing on the opportunity to challenge their methods with activities related to the SARS-CoV-2 virus and the disease it causes, and hoping to make a contribution to tackling it. The range of activity is huge and we can mention only a few examples here.

COVID-19 is unmatched in recent history in terms of the devastation it is causing, both economically and in terms of human life and health.

One area of science in which computation has the potential for impact on COVID-19 research is biomolecular simulation and computational biophysics. These fields use molecular models to study the structures, interactions, and dynamics of proteins and other biological macromolecules. This includes atomically detailed simulations of the components of the SARS-CoV-2 virus and its interactions with host proteins and neutralizing antibodies. Such simulations can help to reveal how viral proteins function, to explore the dynamics of its RNA genome and interactions with protein components, as well as be used to explore the effects of genetic variations (i.e., mutations that the virus adopts during spread). Molecular simulations and related techniques can also potentially contribute to the search for drugs and vaccines. These computational experiments rely on data generated from experimental biological and chemical methods, in particular X-ray crystallography and cryoelectron microscopy (cryoEM), which give highly detailed three-dimensional structural data of the viral machinery and RNA genome. Simulations can provide atomically detailed insight—in particular on protein dynamics—not readily achievable by experiment alone. A so-called force field describes the interactions between the atoms in the system, which may number several million and contain either protein, RNA, DNA, lipids, or a combination of these. The derivative of this interaction potential defines the forces on the atoms, which are numerically encoded and predict their motion, determined by integrating Newton's equation of motion over time. The integration is performed billions or trillions of times at short (femtosecond) time-steps, to build up a trajectory over time of the protein's atomic-level movements. Depending on the size and complexity of the system studied, these calculations can contain hundreds-of-thousands to hundreds-of-millions of atoms, and can run a simulated timescale of nanoseconds to milliseconds. Simulations of this scale are "compute hungry" and with appropriate code, can scale too many hundreds of nodes, thus are often ideally suited to HPC architectures.

The first structures of SARS-CoV-2 proteins appeared in *bioRxiv* (a preprint server, in which researchers disclose scientific results before peer-review and publication in a scientific journal)† in mid-February. The increased adoption of the preprint servers for researchers working on SARS-CoV-2 means that data are coming to light much sooner than waiting for publication in a traditional peer-reviewed journal. For molecular simulation, this is a game changer: such data are particularly important for biomolecular simulations as they generally require structural data as starting points. For example, the first cryoEM

*https://wellcome.ac.uk/coronavirus-covid-19/open-data

†https://www.biorxiv.org

structure of the SARS-CoV-2 spike protein was published in *Science* on March 13, 2020, but the structures it disclosed were made available for researchers via the Protein Data Bank at the time of the preprint's deposition into the *bioRxiv* on February 15, 2020. Early access to these data enabled biomolecular simulation researchers to start working with the structure at least one month sooner than they otherwise would have. Similarly, crystallographers rapidly solved and shared the structure of the SARS-CoV-2 main protease, an enzyme that chops up viral polypeptide to make the proteins the virus needs to assemble in cells. The Protein Data Bank now contains many structures of several viral proteins (solved by groups across the world), including structures bound to small molecules that may be useful in the search for new drugs.

Another shift in research culture has been increased interaction via social media, such as Twitter, which together with preprint servers, webinars and video meetings, are helping to connect scientists across the globe working on this grave threat. News of manuscripts, data, and preprints quickly spreads worldwide. Virtual lab meetings and conferences held over Zoom, WebEx, Skype, BlueJeans, Google Meet, Microsoft Teams, etc., have taken hold and suddenly the global research community has been rapidly connected in new ways. This has helped to compensate for the cancellation and postponement of physical scientific meetings and conferences. The increased spread of information, data sharing, and discussion through emerging communication mechanisms continues to help drive knowledge generation, links between research communities and scientific discoveries about the virus and the complex disease that it causes.

The biomolecular simulation community recognized that, in order to have maximum impact for COVID-19, changes to standard practices would be needed.

In common with other scientific fields. An outcome of this realization is the commitment made by over 200 biomolecular simulation groups, from many countries, to a set of shared principles to share models and data. These principles include using preprint servers to communicate models and results quickly, and sharing methods and data much more quickly than would typically happen in normal scientific publication.[1] Early discussion of methods and data sharing within the simulation community led to the development of a collective site for sharing methods and data through an international joint effort by the US NSF Molecular Software Sciences Institute and European Union BioExcel project.[‡]

Recognizing the potential of computational science—spanning domains from epidemiology to data science to aerosol modeling—governments, research funders and agencies, computer centers and companies have prioritized COVID-19 applications, providing expansive access to HPC and other resources. Several initiatives have been created to support biomolecular simulation across the world (e.g., through PRACE in the EU, ARCHER via the UKRI/EPSRC and the HECBioSim and CCP-BioSim networks in the U.K., RIKEN in Japan, and cloud resources specifically donated by cloud providers such as AWS, Oracle, Microsoft Azure, and Google). Companies such as DE Shaw Research have carried out simulations of viral protein targets and made the results freely available. The folding@home project brought together donated resources worldwide to simulate the products of the viral genome.[2] In the U.S., the COVID-19 High Performance Computing Consortium[§] brings together the most powerful compute resources and is making them broadly available via a rapid proposal process to researchers with appropriate compute needs. What started initially as a consortium in the U.S. quickly spread to international partnerships, including with the United Kingdom and Sweden, making available over 485 petaflops together with technical expertise in software development and other resources. A key aspect of this Consortium is that it provides a mechanism for researchers to get fast access, with application review on the order of days, to support COVID-19 projects. Projects are also working to combine simulations with other types of data and modeling. An example in the EU is Fenix, which is a distributed e-infrastructure providing different types of compute and storage resources. It is being

[‡]https://covid.molssi.org
[§]https://covid19-hpc-consortium.org

used by several different projects performing COVID-19 related research. Some of them are using the HPC resources for simulations. The infrastructure is also being used for sharing data through a publicly accessible object store. Resources are being allocated through different mechanisms including the PRACE Fast Track Call for COVID-19 related projects.¶

With support such as this, efforts of the biomolecular simulation community are contributing to understanding many facets of SARS-CoV-2. All of the proteins of SARS-CoV-2 are the targets of intensive modeling and simulation efforts, by many groups across the world. Similarly, many groups are investigating human proteins that may be involved in the disease. Simulations are especially valuable in augmenting and extending experimental data. A particularly relevant example pertains to the sugary coating that the virus uses to mask its main infection machinery (the "spike" protein) from the human immune system. This so-called "glycan shield" is known to exist, with a particular composition of sugar types, but it is not possible to appreciate what the sugary shield looks like because of experimental limitations. Specifically, the glycans move too much to be captured in static images with high resolution; in other words, we know from experiments that the glycans are present, but scientists cannot "see" the full structure. Molecular dynamics simulations are able to characterize the glycan shield and show how it hides the protein from the immune system (see Fig. 1).[3–5] Simulations are revealing how parts of the spike emerge from this shield to bind to human proteins to infect cells. Simulations are also being used to investigate the effects of genetic variations in the spike that have been identified by experiments. Knowing what parts of the virus surface are exposed, in what circumstances, and which parts of the virus are protected by this coating, allow researchers to understand better how neutralizing antibodies may work. Understanding the exposed portions of the spike may help in the rational design and development of vaccines. A number of efforts are directed at understanding which parts of the virus will lead to B- and T-cell epitopes and present new methods to do so.[7] Simulations may also help identify parts of the spike structure, and the human proteins with which it interacts, that could be targets for binding of small molecule therapeutics.

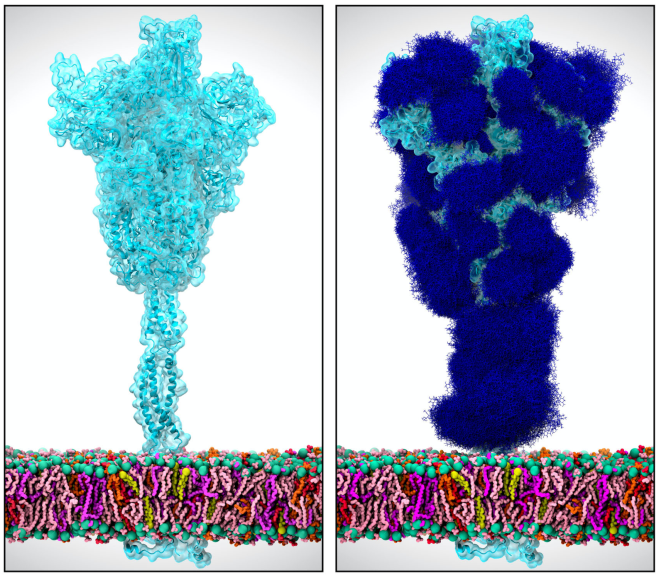


Figure 1. Simulations of the SARS-CoV-2 spike protein, embedded in a viral membrane (pink and purple lines), are being used to inform scientists what the spike looks like with (right panel) and without glycans (right panel, dark blue lines), in order to understand where neutralizing antibodies or drugs may bind.

Biomolecular simulations can help to develop and explore experimentally testable hypotheses. They can potentially be performed rapidly and so "get ahead" of experiments, e.g., delivering early predictions. Simulations can also analyze the effects of genetic variations on the structures and interactions of viral proteins. Computation can test hypotheses, in some cases more rapidly, more cheaply, and on a larger scale than experiments. Reverting to the example of the SARS-CoV-2 spike protein's glycan shield, Casalino *et al.* predicted—ahead of any experimental data—that two glycans near the top of the spike head not only shielded the viral protein but also acted as a molecular trigger that would "lock and load" the spike for infection. This hypothesis was developed and tested *in silico*, and several months later, experimentally confirmed by two independent groups. At the time of writing of this document, several other predictions are being developed and vetted by simulation groups studying SARS-CoV-2 targets as well as their interactions with host proteins. Molecular modeling is helping to analyze recently identified

¶https://fenix-ri.eu/news/using-fenix-resources-covid-19-research interaction of the spike with human neuropilin cell surface receptors[6] as well as potential interactions with the nicotinic acetylcholine receptor.[7] Computing in the age of COVID-19 is providing a plethora of opportunities for simulators to work in concert with experimentalists directly, or indirectly via experimentally-testable predictions disclosed in preprints.

Molecular simulations also have the potential to contribute to the search for possible drugs. A common approach in structure-based drug development is to use *in silico* virtual screening methods to screen vast digitized libraries of small molecule compounds against targets of interest. "Docking" codes can scan large numbers of small molecules to test whether they are likely to bind to a protein target, such as the SARS-CoV-2 main protease. Virtual libraries may contain tens-of-millions to billions of compounds, far more than can be tested experimentally. Researchers are using molecular docking codes to identify potential binders ("hits") among these large numbers of compounds, aiming to provide experimental labs with prospective compounds for testing. For COVID-19, identification of drugs that have been approved for other conditions (drug repurposing), or compounds close to the clinic, that may have activity against SARS-CoV-2 targets (or human protein targets involved in infection or disease pathology) is an attractive approach to finding treatments that could quickly be tested in the clinic. Docking methods are highly approximate and of limited accuracy, so can be combined with more rigorous and detailed molecular simulations to include, e.g., the effects of protein dynamics and to filter out false positives. Due to the large size of compound libraries and depending on the virtual screening method employed, these studies also can utilize HPC architectures.[8] A remarkably early study at Oak Ridge National Lab carried out a large virtual screening campaign of FDA approved compounds to look for potential drug repurposing opportunities and the authors made the predicted results available on the *bioRxiv* in late February, only days after the spike structure was made available on *bioRxiv*. Since then, many similar studies have made early predictions available via this route. Simulations are contributing significantly to the Covid Moonshot Project, closely coordinated with experimental structural work and biochemical tests, in an effort to identify novel drug leads.[9]

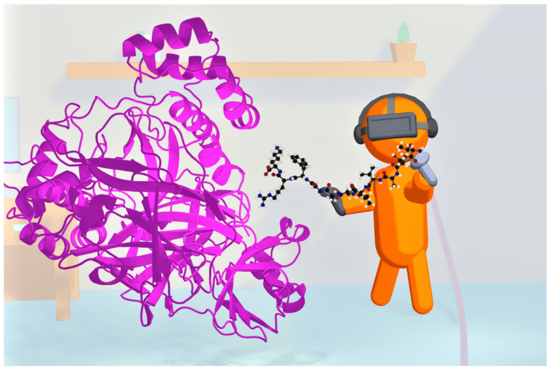


Figure 2. Virtual reality is emerging as a tool to interact with, and manipulate biomolecular simulations. Interactive molecular dynamics simulation in virtual reality (iMD-VR) has the potential to contribute to structure-based drug design, studies of protein structure and function, and education. This cartoon depicts a user "docking" an oligopeptide substrate into the SARS-CoV-2 main protease (magenta), to model how this viral enzyme binds the peptides that it breaks down as part of the COVID-19 viral lifecycle. The flexibility and atomically detailed interactions afforded by iMD-VR allow the user to manipulate the molecular structures to create realistic models.

Simulations can also contribute to understanding other aspects of viral proteins and their mechanisms, which may also help in developing drugs. For example, as mentioned above, the SARS-CoV-2 main protease is a viral enzyme that breaks down long viral polypeptides into pieces that form viral proteins—essential for the manufacture of virus particles in the cell. Understanding the chemical mechanisms by which it does so, and its specificity for particular protein sequences, may help. Chemical reactions in proteins can be simulated by multiscale methods such as combined quantum mechanics/molecular mechanics (QM/MM) techniques.[10,11] QM/MM methods can also be used to predict the reactivity of potential covalent binders as inhibitors and drug leads. Emerging artificial and machine learning methods will also be useful in extracting information from simulation data and connecting with experiment in the search for treatments. Interactive molecular dynamics simulations in virtual reality (iMD-VR, Figure 2) are a new way to interact with and manipulate biomolecular simulations. iMD-VR is an exciting frontier in structure-based drug design. An early example involves small molecule docking with iMD-VR to the SARS-CoV-2 main protease: the combination of 3-D, immersive perception for the use with the flexibility allowed by MD allows peptide substrates to be docked into the enzyme.[12] VR also offers huge potential for data sharing and distributed collaboration: when iMD-VR is cloud mounted, researchers in different physical locations can share the same virtual molecular environment, interacting together with an atomically detailed simulation and model, e.g., a drug binding to its protein target. This could transform how scientists collaborate, allowing researchers to work together directly even when based far from each other, to share and interrogate biomolecular models.

The response of the biomolecular simulation community—from academic groups to supercomputing centers, cloud providers and even chip developers such as NVIDIA—has been impressively strong and rapid, and in many cases, coordinated. Such efforts are already providing new insights and knowledge about the fundamental biology of SARS-CoV-2 as well as contributing to the discovery of novel chemical agents that could be developed into viable therapeutics. Equally striking is how COVID-19 has the potential to catalyze longer term change within the biomolecular simulation community, including the broad adoption of preprint servers for rapidly disclosing research results, and the rapid sharing of methods, models, and data, to disseminate information and knowledge, to test significance and reproducibility of models, and to bring simulation methods and results to other research communities, linking to other areas of scientific investigation to tackle this global pandemic.

ACKNOWLEDGMENTS

We thank Dr. Lorenzo Casalino (UC San Diego) and Becca Walters (University of Bristol) for creating Figure 1 and Figure 2, respectively. We also thank our research groups, as well as collaborators in the work described. We also thank especially Prof. Elisa Fadda (Maynooth University) and her research group, Prof. Michael Feig and Dr. Lim Heo (Michigan State University), Prof. Syma Khalid (University of Southampton), Prof. Carlos Simmerling and his research group (SUNY Stony Brook), Prof. Ben Neuman (Texas A&M University), Prof. Greg Voth and Dr. Viviana Monje-Galvan (University of Chicago), Prof. Julien Michel (University of Edinburgh) and his research group, Prof. Jean-Philip Piquemal (Sorbonne University), Prof. Alessio Lodola (University of Parma), Prof. Vicente Moliner and Dr. Katarzyna Swiderek (Universitat Jaume I), Prof. Garrett Morris and Prof. Fernanda Duarte (Oxford University), Prof. Paolo Carloni (Jülich) and Dr. Reda Rawi (NIH Vaccine Research Center) for helpful discussions. Rommie E. Amaro thanks the Texas Advanced Computing Center for support and assistance with using Frontera. Adrian J. Mulholland thanks the Advanced Computing Research Centre, University of Bristol, the Isambard GW4 EPSRC Tier 2 HPC Centre, HECBioSim/EPSRC and Oracle for Research for computer time and technical support. The work of Rommie E. Amaro was supported by NIH GM132826, NSF RAPID MCB-2032054, an award from the RCSA Research Corp, and a UC San Diego Moore's Cancer Center 2020 SARS-COV-2 seed Grant. The work of Adrian J. Mulholland was supported by EPSRC (CCP-BioSim, Grant number EP/M022609/1), the British Society for Antimicrobial Chemotherapy (Grant number BSAC-COVID-30) and the Elizabeth Blackwell Institute for Health Research, University of Bristol.

REFERENCES

1. R. E. Amaro and A. J. Mulholland, "A community letter regarding sharing biomolecular simulation data for COVID-19," *J. Chem. Inf. Model.*, vol. 60, pp. 2653–2656, 2020, doi:10.1021/acs.jcim.0c00319.
2. M. I. Zimmerman *et al.*, "Citizen scientists create an exascale computer to combat COVID-19," 2020, *arXiv:2020.06.27.175430*.
3. L. Casalino *et al.*, "Beyond shielding: The roles of glycans in the SARS-CoV-2 spike protein," *ACS Central Sci.*, doi:10.1021/acscentsci.0c01056.
4. P. Zhao *et al.*, "Virus-receptor interactions of glycosylated SARS-CoV-2 spike and human ACE2 receptor," *Cell Host Microbe*, to be published, doi:10.1016/j.chom.2020.08.004
5. B. Turoňová *et al.*, "In situ structural analysis of SARS-CoV-2 spike reveals flexibility mediated by three hinges," *Science*, vol. 80, 2020, Art. no. eabd5223. doi:10.1126/science.abd5223.

6. J. L. Daly *et al.*, “Neuropilin-1 is a host factor for SARS-CoV-2 infection,” 2020, *arXiv:2020.06.05.134114*.
7. A. S. F. Oliveira *et al.*, “Simulations support the interaction of the SARS-CoV-2 spike protein with nicotinic acetylcholine receptors and suggest subtype specificity,” 2020, *arXiv:2020.07.16.206680*.
8. M. Smith and J. C. Smith, “Repurposing therapeutics for COVID-19: Supercomputer-based docking to the SARS-CoV-2 viral spike protein and viral spike protein-human ACE2 interface,” *ChemRxiv*, doi:10.26434/chemrxiv.11871402.v3.
9. J. Chodera, A. A. Lee, N. London, and F. von Delft, “Crowdsourcing drug discovery for pandemics,” *Nature Chem.*, vol. 12, 2020, Art. no. 581, doi:10.1038/s41557-020-0496-2.
10. K. Arafet *et al.*, “Mechanism of inhibition of SARS-CoV-2 Mpro by N3 Peptidyl Michael acceptor explained by QM/MM simulations and design of new derivatives with tunable chemical reactivity,” *ChemRxiv*, doi: 10.26434/chemrxiv.12941819.v1.
11. C. A. Ramos-Guzmán, J. J. Ruiz-Pernía, and I. Tuñón, “Unraveling the SARS-CoV-2 main protease mechanism using multiscale DFT/MM methods,” *ChemRxiv*, doi:10.26434/chemrxiv.12501734.v2.
12. H. M. Deeks, R. K. Walters, J. Barnoud, D. R. Glowacki, and A. J. Mulholland, “Interactive molecular dynamics in virtual reality (iMD-VR) is an effective tool for flexible substrate and inhibitor docking to the SARS-CoV-2 main protease,” *ChemRxiv*, doi: 10.26434/chemrxiv.12834335.v2.

Rommie E. Amaro received the Distinguished Professorship degree in theoretical and computational chemistry from the Department of Chemistry and Biochemistry, University of California, San Diego, San Diego, CA, USA, the B.S. degree in chemical engineering and the Ph.D. degree in chemistry from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 1999 and 2005, respectively. She was an NIH NRSA Postdoctoral Fellow with Prof. J. Andrew McCammon with UC San Diego from 2005 to 2009, and started her independent research program in 2009 with the University of California, Irvine, CA, USA. In 2012, she moved her lab to the Department of Chemistry and Biochemistry, UC San Diego. She is the recipient of an NIH New Innovator Award, the Presidential Early Career Award for Scientists and Engineers, the ACS COMP OpenEye Outstanding Junior Faculty Award, the ACS Kavli Foundation Emerging Leader in Chemistry National Lecturer, and the Corwin Hansch Award. Her scientific interests lie at the intersection of computer-aided drug discovery and biophysical simulation. Her scientific vision revolves around expanding the range and complexity of molecular constituents represented in such simulations, the development of novel multiscale methods for elucidating their time dependent dynamics, and the discovery of novel chemical matter controlling biological function. Contact her at ramaro@ucsd.edu.

Adrian J. Mulholland is currently a Professor of Chemistry with the University of Bristol. His research focuses on investigating biomolecular structure and function using molecular and multiscale simulation, in applications ranging from drug resistance to biocatalysis. Following his first degree at Bristol, he worked in a wine merchant and then for ICI (now AstraZeneca) Pharmaceuticals before his doctoral studies with Prof. Graham Richards (Oxford University). He was then a Wellcome Trust Prize Fellow with Prof. Martin Karplus (Harvard University) before returning to take up a position in Bristol, where he is currently the Director of Research with the School of Chemistry. He has held Fellowships from EPSRC and the Wellcome Trust, and was founding Chair of CCP-BioSim (the U.K. Collaborative Computational Project on Biomolecular Simulation) and HECBioSim (the U.K. High End Computing Consortium for Biomolecular Simulation). He was elected Chair of the Molecular Graphics and Modeling Society, and of the 2016 Gordon Research Conference on Computational Chemistry. He was the inaugural Lakshmi Raman Lecturer (University of Pittsburgh, 2019) and is a recipient of a Wiley/ISQBP Young Investigator Award and the 2020 John Meurig Thomas Medal for outstanding and innovative work in catalytic science. Contact him at Adrian.Mulholland@bristol.ac.uk.

Theme Article: Computational Science in the Battle Against COVID-19

Cloud Computing for COVID-19: Lessons Learned From Massively Parallel Models of Ventilator Splitting

Michael Kaplan
Duke University School of Medicine

Charles Kneifel, Victor Orlikowski, James Dorff, and Mike Newton
Duke University Office of Information Technology

Andy Howard
Microsoft

Don Shinn
CrossComm

Muath Bishawi, Simbarashe Chidyagwai, Peter Balogh, and Amanda Randles
Duke University

***Abstract*—A patient-specific airflow simulation was developed to help address the pressing need for an expansion of the ventilator capacity in response to the COVID-19 pandemic. The computational model provides guidance regarding how to split a ventilator between two or more patients with differing respiratory physiologies. To address the need for fast deployment and identification of optimal patient-specific tuning, there was a need to simulate hundreds of millions of different clinically relevant parameter combinations in a short time. This task, driven by the dire circumstances, presented unique computational and research challenges. We present here the guiding principles and lessons learned as to how a large-scale and robust cloud instance was designed and deployed within 24 hours and 800 000 compute hours were utilized in a 72-hour period. We discuss the design choices to enable a quick turnaround of the model, execute the simulation, and create an intuitive and interactive interface.**

Digital Object Identifier 10.1109/MCSE.2020.3024062
Date of publication 21 September 2020; date of current version 9 October 2020.

The ability to quickly spin-up cloud instances that strategically match the presented research question, alongside the increasing availability of high-performance computing resources, has resulted in an increased use of cloud computing for critical computational science research. A global pandemic, such as COVID-19, naturally amplifies the need for rapid development and deployment of tools that can have an impact. The dire circumstances evolving in spring of 2020 led to increased hospitalization rates and estimates of shortages of ventilators within the U.S. alone of between 45 000 and 160 000.[1] Ventilators are an essential component of life-preserving treatment for patients with respiratory failure and a shortage of ventilators was predicted,[2] and in some cases realized,[3] early in the global pandemic. As a result, considerable effort has been focused on methods to share one ventilator amongst multiple patients.[1,4,5] However, due to safety concerns[6] with preexisting ventilator splitting strategies, ventilator splitting in the past has not been recommended.[7] In collaboration with restor3D (a local biotechnology company, https://www.restor3d.com/), a large team of engineers and clinicians at Duke University developed a ventilator splitter and resistor system (VSRS) aimed at increasing the safety profile of ventilator splitting by accurately predicting the delivered tidal volumes and pressures under the wide range of clinically relevant situations. To achieve this task, it was necessary to quickly develop and deploy new strategies of ventilator splitting, combining 3-D printed components designed to fit standard ventilator tubing (see Figure 1: top) with extensive computational modeling to ensure that each patient would receive a safe degree of ventilation (see Figure 1, bottom).

With the validated device in hand, the remaining question was how to tune it for any given patient pairing and, further, how to provide this decision support in an intuitive way to the clinicians. Addressing these questions presented unique challenges in an environment with significant time pressure. The entire process, from design of a new computational model, to architecting an appropriate computing environment, to creating a user interface, presented many challenging opportunities of interest to high-performance computing and biomedical researchers facing problems driven by short turnaround times. This article summarizes our design decisions and lessons learned during model design, HPC resource procurement and deployment, and execution of a massively parallel solution using over 800 000 compute hours in a 72 h period. The total time to create the initial model, validate it against benchtop data, deploy the model at scale, and to collate the simulation results into an easy-to-use mobile app was less

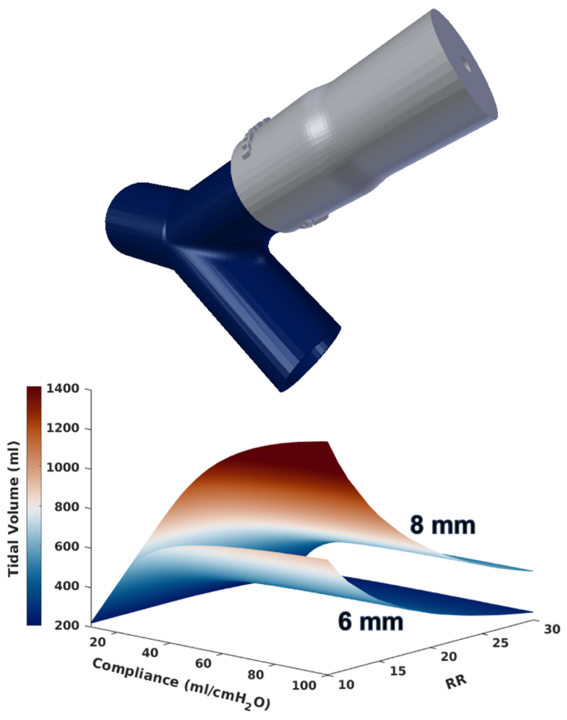


Figure 1. *Top:* Three-dimensional (3-D) printed splitter (in blue) connected to the resistor (silver). Airflow from the ventilator comes into the VSRS from the left and heads to the patient who requires the resistor (top right) or without the resistor (bottom right). *Bottom:* Example output from the numerical model without a resistor for a PIP of 30, PEEP of 8, and I:E of 1 (acronyms explained in Table 1), demonstrating how predicted tidal volumes vary greatly and nonlinearly from multiple parameters, specifically pulmonary compliance, respiratory rate, and endotracheal (ET) tube size. The result for two different ET tube sizes are displayed, with an 8 mm ET tube resulting in greater tidal volumes than a 6 mm ET tube. Red colors demonstrate potentially highly dangerous tidal volumes.

than one month. We expect that these experiences can be applied by the computational science community to future research questions requiring rapid deployment.

DESIGN: MODEL DEVELOPMENT

Establishing a Numerical Model

A numerical model was established to aid clinicians in their use of the VSRS, by calculating the airflow characteristics and distribution through the system over a wide range of operating conditions. Under the dire circumstances of a global pandemic, not only was time-to-solution a driving design goal, but similarly was minimal development time. In conventional computational fluid dynamics research, time would be taken to optimize the code and the simulation setup so that the run-time of each simulation was minimized. Such an effort typically requires significant man-hours to establish a validated and optimized computational model. The challenges in developing the VSRS model were the competing needs of 1) creating a robust, well-validated, and accurate model to ensure that clinicians can deploy the VSRS while minimizing harm to their patients, and 2) the need to deliver results as quickly as possible.

Due to a lack of well-established numerical models available for ventilator splitting, we developed our own model to explore the general dynamics of a system where multiple patients are connected to one ventilator. Specifically, we approached the problem by using lumped parameter models to solve the governing equations of mass, energy, and momentum conservation in order to simulate airflow from a ventilator source to a patient's lungs. The lungs were modeled as a Hookean spring and viscous dashpot in parallel to represent the pulmonary compliance and resistance, respectively. The relevant inputs to the model are described in Table 1 and Figure 2, and the outputs are time-series of delivered tidal volumes (which can be condensed as displayed in Figure 1, bottom) and pressures.

The ideal modeling tool for this task would be one which can be run locally, is highly flexible to allow different configurations and patient parameters, and is readily scalable and computationally efficient so that it could eventually be deployed for a massive parameter sweep. MATLABs Simscape has an easy-to-use GUI, which allows for rapid generation of highly flexible models that can run easily on a laptop. This allows one to experiment with the significance of different parameters and model configurations, however, using a higher-level proprietary software that requires a license to run potentially meant that we would encounter issues with scalability and efficiency when deploying the model at scale. While we chose MATLAB due to previous experience with the modeling software in an effort to decrease time to model generation, open source alternatives, such as OpenModelica or Scilab, could have been explored as well.

The initial models helped us make fundamental observations, which were key to designing our large-scale parameter sweep. Of note, we discovered that the one ventilator mode (pressure-controlled ventilation) was inherently safer for ventilator splitting than another (volume-controlled ventilation). Additionally, the preliminary models demonstrated that a decoupling occurs when multiple patients are placed on pressure-

Table 1. Input parameters to the numerical model.

Parameter Name	Units	Min Value	Max Value	Step Size	# of Values
Peak Inspiratory Pressure (PIP)	cmH_{2O}	20	50	1	31
Positive End-Expiratory Pressure (PEEP)	cmH_{2O}	5	20	1	16
Inspiratory to Expiratory Ratio (I:E)	ratio	1:3	1:1	fraction	7
Respiratory Rate (RR)	$\frac{\text{breaths}}{\text{min}}$	10	30	1	21
Pulmonary Compliance	$\frac{\text{ml}}{\text{cmH2O}}$	10	100	1-2	46-91
Endotracheal Tube (ET) Diameter	mm	6	8.5	0.5	6
Resistor Radii	mm	2.5	5.5	1	7

Final parameters, along with the minimum, maximum, and step size, and discrete values explored, that were required for the large parameter sweep. Step sizes were determined so that an incremental change resulted in less than a 5% change in delivered tidal volumes. The top four parameters are ventilator settings (PIP, PEEP, I:E, RR), the next two are patient-specific considerations (Compliances and ET tube size), and the final one is a circuit configuration parameter (Resistor size).

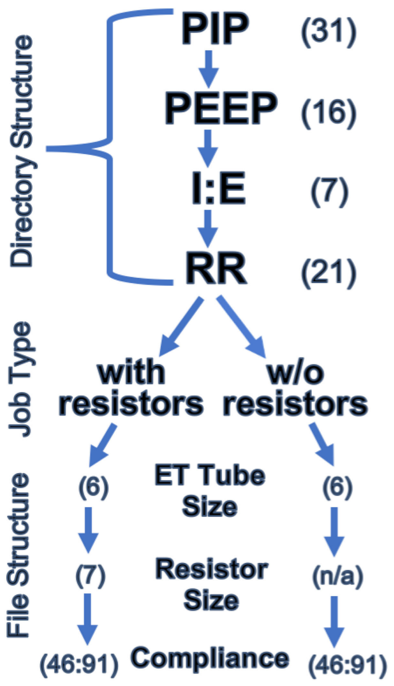


Figure 2. Illustration of the directory structure, job types, and output file structure for the 7-D embarrassingly parallel parameter sweep. The number of parameters at each level are in parentheses. For each combination of PIP, PEEP, I:E, and RR, two jobs were submitted, for the cases without a resistor and one for cases with a resistor. Those jobs then swept through the parameters of ET Tube sizes, resistor sizes (if applicable), and pulmonary compliances, totaling over 270 million different simulations.

controlled ventilation, such that there is no interaction between patients connected to the same ventilator. This finding was important because it fundamentally changed our computational task. Namely, instead of simulating every possible combination of ventilator parameters as well as both patients' individual characteristics, this permitted the more tractable task of solving for every possible combination of ventilator parameters and a single patients characteristics.

This difference ultimately reduced the number of simulations we had to perform by several orders of magnitude (270 million as compared to 125 billion) and turned an intractable task into a large, but achievable computational challenge given the time constraints.

A final important outcome from the initial testing was to understand the sensitivity of our outputs of interest, which are tidal volumes (see Figure 1 bottom) and delivered pressures, to the many input parameters (see Table 1). The minimum and maximum values for various parameters were known from clinical experience, but how finely we would have to sample parameter space was unknown. To overcome this, we performed sensitivity tests to determine the granularity with which we would have to sample parameter space in order to guarantee precise results within an acceptable level of uncertainty. With this knowledge, and initial speed tests of the numerical model on local resources, we were able to estimate the number of simulations and compute hours that would be required, assuming that we were able to efficiently scale the model to larger systems.

Lesson Learned 1: *Using low overhead tools, such as MATLAB's Simscape, for initial modeling can enable rapid acquisition of baseline intuition needed for design of the large-scale study.*

Embarrassingly Parallel Design

The goal of this project was to provide a decision support tool to aid clinicians in their use of the VSRS. To this end, we needed to precalculate the expected airflow for any potential patient- and ventilator-derived values, so that the resulting data could be made available in realtime to help direct the choice of resistor. As described in Table 1, there are seven parameters serving as input to the simulation. For the four ventilator settings in Table 1 (PIP, PEEP, I:E, RR) and ET tube diameters, the step size was chosen to allow clinicians to have an equal level of granularity as they would with a standard ventilator, with minimum and maximum values chosen based on clinical experience for the range of realistic clinical scenarios. The compliance step size was set such that a step change in compliance would lead to less than a 5% change in tidal volumes. Determining the number of values to explore was derived from discrete subtraction of the minimum from the maximum value, divided by the step size.

Figure 2 displays the chosen hierarchy for the parameter sweep. The batch submission script generates two jobs (one for the model with a resistor, one for the model without a resistor) for each combination of PIP, PEEP, I:

E, and RR. Each job then sweeps through the different ET Tube sizes, Resistor sizes, and Compliance values. Due to the embarrassingly parallel nature of the parameter sweep, it was possible to break up the number of simulations into jobs in a variety of different ways. Ultimately, a balance was chosen such that an average job would last approximately 5 h and, therefore, resubmitting a failed job would not incur an undue strain on the allotment. Additionally, this results in a manageable number of writes to disk, with a tolerable overhead cost for each job.

With a cloud-based environment different node counts can be running, and generally speaking there is a lot of flexibility. This is well suited for an embarrassingly parallel framework. We further optimized the design by having each simulation save the time-series data for pressures and tidal volumes to the local disk. This was facilitated by a hierarchy that identified where in parameter space that simulation occurred, for ease of postprocessing and collating the results. In order to not overwhelm the storage capacity on-node, prior to completion the job would postprocess the results by: 1) determining the tidal volume, maximum pressures, and minimum pressures at the steady state, 2) delete the time-series data, and 3) create a reduced precision csv file with a row for each simulation that was included in the job. This configuration was designed to allow for easy debugging and resilience against network outages. Namely, one could look at the files on-node to determine which specific simulation failed by finding the last successful output in a parameter sweep. In total 146 000 intermediate csv files were created, for which subsequent scripts were deployed to concatenate them along the directory structure until the final data table was ultimately produced.

Lesson Learned 2: *Relying on an embarrassingly parallel framework allowed us to match to a dynamic cloud-based environment, to best facilitate the required large-scale parameter sweep.*

DEPLOYMENT: DYNAMIC CLOUD COMPUTING

With the initial modeling was complete, the next step was to develop a method for deploying the model at scale. In this section, we will discuss the unique opportunities provided by deploying on a cloud platform, unexpected challenges with maintaining an embarrassingly parallel code at scale, solutions we implemented, and how the hundreds of millions of simulations were synthesized into an intuitive and portable interface for clinicians.

Problem-Oriented Architecture Design

To obtain access to high-performance computing resources on a scale similar to what was required for this project, it is common practice to submit an architecture-targeted proposal requesting compute hours on a specific resource. While drafting such a proposal, care is taken to demonstrate feasibility, run-time, and parallel performance on the targeted system, since efficiency is paramount on these precious resources. For our project, however, the emphasis was the need for haste in solving the problem, which led to us seeking methods that were almost agnostic to the underlying hardware. The COVID-19 HPC Consortium (https://covid-19-hpc-consortium.org/) provided a unique opportunity for describing the needs and constraints posed by the problem, which could then be paired with an appropriate and available resource allocation. In this case, we had a working, validated code that could easily run on a variety of different platforms. The embarrassingly parallel nature of the setup meant that network connectivity was less of a consideration, and that our primary need was a large core count with sufficient memory (2–4 GB of memory-per-core) so that each core could independently complete a simulation without being bottlenecked by shared memory. Most importantly, we needed access to a high-throughput resource, where jobs could be submitted and executed quickly. Although time-to-solution was important, the constraints allowed some flexibility. For example, individual job run-time was not a concern, nor was the order of completion; if some jobs took longer to run than others, that was acceptable. The overriding need was to turnaround completion of all of the jobs within a few days so that the data were available for FDA review of the device and associated clinical support software. Rather than requesting a set number of core hours corresponding to jobs on set architecture and node sizes, the Consortium afforded us the ability to describe the problem, needs, and constraints. This flexible, problem-oriented design process facilitated a more efficient and effective matching of resources to problem that played a critical role in our ability to successfully accomplish our stated research goals.

The shift to a problem-driven approach was further enhanced through being paired with the Microsoft team for computing resource coordination and administration. The Consortium provided resources through the Microsoft cloud computing infrastructure, Azure. The use of a cloud-based architecture was a strong fit for our needs based on the ability to configure the infrastructural aggregate to suit the problem at hand. Thus, as we were leveraging an embarrassingly parallel setup, there was no necessity for the node count to remain fixed over the course of the simulations. Throughput, resilience, and resource availability could be balanced in a dynamic way as node counts assigned to the jobs could fluctuate throughout the execution duration. In the initial meetings between the Duke and Microsoft teams, we were able to outline the problem and associated resource needs to allow a cloud instance to be configured and tuned specifically to meet our need. As mentioned, we opted for an implementation that left each parameter-based simulation contained in an independent manner with minimal data collection and analysis handled between small groups of tasks. Therefore, rather than a tightly coupled, MPI-based model, we were able to employ a minimal communication, embarrassingly parallel framework. This design choice resulted in the processor selection being of far more consequence than choice of interconnect, so we searched for an Azure cloud configuration, which would allow for the largest core count at the lowest cost per core. The goal of maximizing throughput and minimizing wall-time could only be accomplished by taking advantage of as many nodes as was economically feasible. Shared storage requirements were relatively minimal, since individual compute nodes did not need to reference considerable shared data and post-processing was designed to be performed on the local node; requirements were limited to those necessary for job submission—Slurm configuration and the users home directory. In order to scale the NFS filesystem on the head node to support hundreds of nodes connecting back to it, a managed premium disk was attached to the head node to use as the NFS export. With these considerations, the team was able to define a rough architecture using the Azure HB-series VMs, a basic NFS filesystem, and a Slurm front-end to manage job scheduling, all orchestrated by Azure CycleCloud. Slurm was chosen as the cloud scheduler to allow a seamless transition from our local cluster, which also used Slurm, to the cloud, yet another benefit of being able to fully customize the cloud architecture to meet the needs of the project. As a result, we were able to rapidly deploy our model with minimal time spent altering the code.

With an architecture defined, the next step was calculating the number of cores needed to meet the timeline. Based on this architecture and the desired timeframe, 24 000 cores were estimated to be needed to complete the project in 2 days (24 000 cores $\times$ 48 h gave a total of 1 152 000 core hours). Alternatively, if we attempted to complete this task with only local resources (for example, by having unrestricted access to 1000 cores), the total time to solution (over a month) would have delayed our ability to combat the initial surge in COVID cases. Due to the computations not requiring communication between compute nodes and the capabilities of Azure, it would have been possible to secure the necessary number of cores by combining allocations from several geographically distinct locations—with some jobs running in Europe, for example, while others might run in South America. In order to simplify the debugging of failed jobs and minimize administrative overhead, however, all nodes were allocated within the same datacenter in Western Europe; thus, 24 000 cores (using the Azure individual node type “HB60s”) were made available by Saturday.

Lesson Learned 3: *By configuring the cloud architecture to match the needs of the problem (a problem-oriented approach) instead of manipulating the problem to fit the constraints of the platform (an architecture-oriented approach), we were able to be prepared to rapidly deploy our model.*

Lesson Learned 4: *By mimicking the feasibility testing environment, and thereby minimizing the* *need to rework scripts, we were able to rapidly implement our model at scale.*

EXECUTION: COMPLETING 800 000 COMPUTE HOURS OF SIMULATION IN 72 H

In order to successfully complete the computing challenge, it was important to anticipate issues before they arose and to design the system architecture accordingly. However, some issues with underlying dependencies and hidden performance bottlenecks only arose when the model was deployed at the full scale. Solving these issues rapidly was paramount to being able to complete the computational task without depleting the allotment of compute hours.

Identify Underlying Dependencies

While initial performance testing on the cloud behaved as expected, significant slowdowns were observed while attempting to spin up all of the nodes and deploy the model. This eventually reached the point where job submission became infeasible and performance levels were significantly below expectations (both in terms of concurrently running jobs and completion rates).

To investigate this, we identified all communication channels, storage locations used, and potential hidden bottlenecks in the architecture and configuration of the supporting software. In this manner, a number of issues were addressed, including network addressing and Slurm scheduler tuning. The most significant issue identified was that file I/O from the head node slowed and eventually stopped. The cause was identified as the shared NFS filesystem being written to by all the running jobs, but only minimally; the I/O load was not commensurate with the slowdown being experienced. The underlying reason was a MATLAB-specific setting associated with the simulation execution, related to the location of the preferences directory. While unrelated to the simulation itself, this emphasizes the importance of running full-scale tests to identify hidden infrastructure interactions.

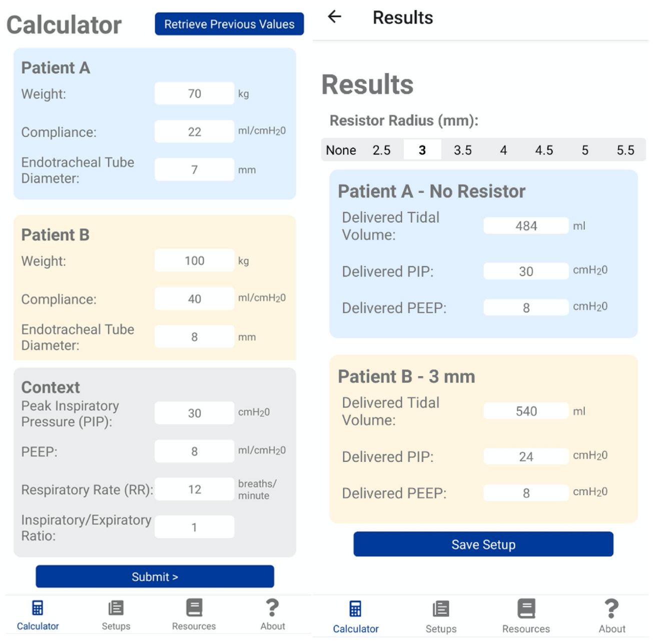
Lessons Learned 5: *Beyond initial performance testing, full-scale tests deploying the model helped to identify hidden dependencies causing severe performance degradation.*

Look for Hidden Performance Bottlenecks

Hidden performance bottlenecks at scale can create significant increases in necessary compute hours. For example, MATLAB is based on run-time compilation, and the time to compile the code was several factors larger than the time necessary to simulate the model for a given set of parameters. As a result, a primary consideration to maintain efficiency was to minimize time spent compiling. Fortunately, there is built-in functionality in MATLAB to perform a parameter sweep without recompiling the code for each iteration of a parameter sweep. This was only possible to implement, however, for sweeps over patient parameters; this was not easily achievable for sweeping through ventilator settings or changes in the circuit architecture. We took advantage of this by organizing the simulations into separate jobs for which only one compilation was required per job. This resulted in the 270 million simulations being handled by 146 000 jobs, which drastically jminimized wall-time by reducing run-time recompilation.

Use of a folder directory architecture (see Figure 2) whereby files are stored across multiple directories, reduced slow-down associated with reading and writing to disk compared with single-directory storage. This was an important consideration given the significant I/O associated with the $O(10^5)$ files associated with the 146 000 jobs. An additional advantage of storing both the submission scripts and output in this separated-directory structure was the triviality of generating auxiliary scripts for detecting failures in either the job submissions or model output generation; this allowed problematic jobs to be easily found and resubmitted.

To decrease data storage and transfer requirements, all postprocessing was performed on-node immediately after each simulation was completed, with intermediate results deleted. This converted the time-series output into three values (the steady-state tidal volume, maximum delivered pressure, and minimum delivered pressure), which were stored only at the clinically relevant precision. As a result, what would have required



Calculator

Retrieve Previous Values

Patient A

Weight:	70	kg
Compliance:	22	ml/cmH20
Endotracheal Tube Diameter:	7	mm

Patient B

Weight:	100	kg
Compliance:	40	ml/cmH20
Endotracheal Tube Diameter:	8	mm

Context

Peak Inspiratory Pressure (PIP):	30	cmH20
PEEP:	8	ml/cmH20
Respiratory Rate (RR):	12	breaths/minute
Inspiratory/Expiratory Ratio:	1	

Submit >

Results

Results

Resistor Radius (mm):

None 2.5 **3** 3.5 4 4.5 5 5.5

Patient A - No Resistor

Delivered Tidal Volume:	484	ml
Delivered PIP:	30	cmH20
Delivered PEEP:	8	cmH20

Patient B - 3 mm

Delivered Tidal Volume:	540	ml
Delivered PIP:	24	cmH20
Delivered PEEP:	8	cmH20

Save Setup

Figure 3. Illustration of the mobile app's input (left) and output (right) displays. Note that the inputs are from a drop-down menu and are not free text fields to reduce the possibility of clinician error. The displayed output is for the case of the user querying the results for a 3 mm resistor.

over 100 TB of storage space was decreased to approximately 10 GB.

Lesson Learned 6: *Job organization with a run-time compilation code like MATLAB is important to minimize recompilation and, thus, maintain the embarrassingly parallel characteristic.*

Lesson Learned 7: *I/O, data storage, and transfers can be improved through specific usage of directory structures and on-node postprocessing.*

TRANSLATION: ESTABLISHING AN INTUITIVE INTERFACE

With simulations completed, the remaining challenge was how to provide this data back to the clinician in a way that is simple, intuitive, up-to-date, and minimizes the chance of error. A mobile app, both for iOS and Android, that can run on low-end mobile phones as well as high-end tablets maximizes portability of the VSRS to global health scenarios as well as high-tech ICUs. Using a mobile app allows for a native, performant user interface (see Figure 3) and the ability to save and retrieve deidentified input value sets locally on the device. As the final data table, which stores all of the precomputed results, is larger than 10 GB, it is unlikely to be easily stored on low-end mobile phones. Consequently, it was decided to have the mobile app connect to a cloud-based API to receive the input values and return the corresponding results from the indexed database. While an Internet connection is necessary to retrieve new results, the installed app approach also leaves the door open to potentially precaching the data on high-capacity mobile devices should a no-Internet version be necessary in the future. An advantage of the cloud-based API is that it allows for clinicians to have the most up-to-date and accurate results at their fingertips. While the VSRS exclusively makes use of a mobile app, a remote webform is a reasonable alternative that could deliver similar functionality as the mobile app.

As the data were precalculated, the end-user would need to query precalculated results based on a diverse but fixed number of input possibilities. Taking advantage of this and with an intent to reduce possible user-error, we followed a selection-based UI paradigm (akin to tabs and drop-down menus). This ensured that clinicians would only be able to input values that were consistent with the precomputed results and that no errors, such as confusion with unit conventions, would occur.

Lessons Learned 8: *A cross-platform mobile app with a selection-based UI maximizes usability in various hospital settings, while minimizing user error.*

CONCLUSION

Multiple lessons were learned in the process of rapid development and deployment of a parallel numerical model to support the clinical use of the VSRS in the event of ventilator shortages. Unlike conventional research projects that are designed and executed over months or years, a unique set of challenges arise for projects requiring rapid and agile development and deployment. The balance of developer time versus compute time under severe time-to-solution constraints leads to substantively different design choices. This article is an attempt to organize and articulate the valuable lessons learned in the process of generating

APPENDIX

Airflow Numerical Model

Air flow was simulated as a lumped parameter model using MathWork's Simscape pipe flow dynamics packages, which solve the laws of mass, momentum, and energy to determine the pressure, velocity, density, and temperature of gas as it travels through a network of pipes. The ventilator was modeled as either a volume or pressure source with a specified waveform representing the user-defined ventilator settings (PIP, PEEP, I:E, RR). Gas flow is then simulated for the travel though standard ventilator tubing, where it then interacts with the splitter, resistor of set diameter, endotracheal tube of a given size, and the patient's lungs. The lungs were modeled as a Hookean spring, representing the inverse of the compliance of the lungs, and a viscous dashpot, representing the resistance of the lungs, in parallel.[8] The range of input parameters to the model were chosen based on simulating the wide range of ventilator settings, endotracheal tube sizes, and pulmonary characteristics that clinicians could encounter when treating patients with respiratory failure. Models were simulated until the gas flow to the lungs reached the steady state. Sensitivity tests were conducted to determine the granularity of the parameter sweep necessary to ensure that a change in tidal volume of less than 5% occurred for a given step size.

The model was validated against benchtop data. Using an anesthesia care station ventilator, standard tubing, the 3-D printed splitter and resistor system, standard endotracheal tubing, and artificial test lungs, experiments were performed to determine the delivered pressures and tidal volumes to the artificial test lungs for different ventilator settings. When these same settings were used as inputs to the numerical model, the predicted tidal volumes were found to be in excellent agreement with those from the benchtop experiments.

the data necessary to guide resistor choice when using the VSRS to split ventilators between two or more patients.

For example, utilizing physical intuition gleaned from preliminary modeling can assist in the development of an embarrassingly parallel numerical model, which is advantageous for performing large-scale parameter sweeps in HPC environments. A cloud architecture as the HPC environment allows for platform customization to match the needs of the problem, instead of having to coerce the problem into functioning on the platform, which leads for rapid deployment. In order to avoid significant slowdowns when an embarrassingly parallel code is deployed at a massive scale, it is important to analyze the code and communication channels for hidden dependencies and performance bottlenecks. Last, for the results of the hundreds of millions of simulations to be utilized to help combat COVID-19, it is essential to create an intuitive user interface, with effort placed to minimize the potential user error. These lessons learned from rapidly deploying almost one million compute hours in a cloud-based infrastructure for a COVID-19 target are applicable to other situations where researchers have maximal motivation to minimize time-to-solution.

ACKNOWLEDGMENTS

The authors would like to thank many contributors who provided assistance, often at odd hours, to help complete this project with a fast turnaround. In particular, they would like to thank the COVID-19 HPC Consortium for both the compute hours and fast turnaround time and the broader Microsoft team for all of the support. They would like to thank T. Milledge for connecting us with the larger Research Computing team at OIT and assisting with MATLAB licensing issues, T. Lonergan for acting as the licensing liaison between Mathworks and Azure, and J. Hopkins and S. Rich for technical assistance with CycleCloud authorization tokens. The team would also like to thank the assistance of Microsoft and the Microsoft team. They provided assistance, insight, and funding all on short notice and with full commitment to the success of the project.

REFERENCES

1. S. Srinivasan *et al.*, “A rapidly deployable individualized system for augmenting ventilator capacity,” *Sci. Trans. Med.*, vol. 9401, no. May, 2020, Art. no. eabb9401, doi: 10.1126/scitranslmed.abb9401.
2. B. Rosenthal and A. Feuer, “Coronavirus in N.Y.: Astronomical surge leads to quarantine warning,” pp. 3–7, 2020. [Online]. Available: https://www.nytimes.com/2020/03/24/nyregion/coronavirus-new-york-apex-a ndrew-cuomo.html
3. J. R. Beitler *et al.*, “Ventilator sharing during an acute shortage caused by the COVID-19 pandemic,” *Amer. J. Respiratory Crit. Care Med.*, vol. 15, pp. 600–604, 2020. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/32515988
4. B. K. Lai, J. L. Erian, S. H. Pew, and M. S. Eckmann, “Emergency open-source three-dimensional printable ventilator circuit splitter and flow regulator during the COVID-19 pandemic,” *Anesthesiology*, vol. 133, pp. 246–248, 2020.
5. A. Clarke, A. Stephens, S. Liao, T. Byrne, and S. Gregory, “Coping with COVID-19: ventilator splitting with differential driving pressures using standard hospital equipment,” *Anaesthesia*, vol. 75, pp. 872–880, Apr. 2020, doi: 10.1111/anae.15078.
6. A. D. Cherry, J. Cappiello, M. Bishawi, M. G. Hollidge, and D. B. MacLeod, “Shared ventilation: Toward safer ventilator splitting in resource emergencies,” *Anesthesiology*, vol. 133, pp. 681–683, Sept. 2020.
7. “Joint Statement on Multiple Patients Per Ventilator,” pp. 12–14, 2020. [Online]. Available: https://www.asahq.org/about-asa/newsroom/news-releases/2020/03/joint-st atement-on-multiple-patients-per-ventilator
8. M. Schmidt *et al.*, *Computer Simulation Measured Respiratory Impedance Newborn Infants Effect Measurement Equipment*, vol. 20. Amsterdam, The Netherlands: Elsevier, 1998, no. 3.

Michael Kaplan is currently a medical student with Duke University, Durham, NC, USA, conducting research in the Biomedical Engineering Lab of Prof. A. Randles. He plans to start residency in Anesthesiology in 2021 and to continue to conduct research in basic and translational sciences, in fields ranging from computational hemodynamic modeling, machine learning, surgical optimization, vascular growth, and imaging of the microvasculature. He received the Ph.D. degree in geophysics from the University of Southern California, Los Angeles, CA, USA. Contact him at mike.kaplan@duke.edu.

Charles Kneifel is currently the Senior Technical Director with Duke OIT, Durham, NC, USA. He manages Dukes central technology infrastructure and Software Defined Networking Project. He has coordinated several technology grants with Duke including the National Science Foundations Data Infrastructure Building Blocks (DIBBS) grant to build campus cyberinfrastructures. Prior to Duke, he was a Chief Information Officer with the American Kennel Club for nine years. He has also held multiple technical positions at the NC State University. He received the Ph.D. degree in chemistry from the State University of New York at Stony Brook, Stony Brook, NY, USA. Contact him at charley.kneifel@duke.edu.

Victor Orlikowski is currently a Research Software Developer and Systems Administrator with Duke Research Computing, Stony Brook, NY, USA. He is a Senior IT Analyst focused on automation. He is experienced in networked storage devices and virtual machines. He has been a key member of research teams that have pioneered the tools regularly used by researchers at Duke. He has been with Duke for more than eight years in the Departments of Computer Science, Pratt School of Engineering and OIT. He regularly teaches Dukes Introduction to Linux seminars. Contact him at vjo@duke.edu.

James Dorff is currently an IT Senior Manager with Duke University, Durham, NC, USA. He specializes in Unix System Administration and computational physics. Contact him at jdorff@duke.edu.

Mike Newton is currently the Senior IT Analyst with Duke Universitys Office of Information Technology (OIT), Durham, NC, USA. He is the Primary System Administrator for OITs High Performance Computing (HPC) cluster. Contact him at jmnewton@duke.edu.

Andy Howard is currently a Senior Program Manager on the Azure Compute team focused on HPC and Azure CycleCloud. After spending almost a decade helping architect, build, and manage on-premises HPC systems for companies such as Purdue University, Cray, and the Department of Defense, he has spent the last seven years helping transition HPC workloads to the cloud. At Microsoft, he draws on this industry experience to help customers plan cloud HPC implementations and guide Microsoft’s HPC product offerings. He received the B.S. degree in electrical and computer engineering technology from Purdue University, West Lafayette, IN, USA. Contact him at Howard.Andy@microsoft.com.

Don Shinn is currently the Founder and CEO of CrossComm—an award-winning mobile, web, and immersive app development studio with a 20+ year history of deploying innovative technologies to solve the toughest problems and challenges. Under Mr. Shin's leadership, the Durham, NC-based company has been recognized as one of the leading mobile and AR/VR app developers in the region by clutch.co, and has been nationally recognized as the Minority Technology Firm of the Year (2015) by the US Department of Commerce. He has been a passionate advocate for human-centric user interfaces throughout his career, and is currently interested in exploring the future of spatial computing and leveraging app innovation to advance social good. Contact him at don.shin@crosscomm.com.

Muath Bishawi is currently a Cardiothoracic Surgery resident with Duke University, Durham, NC, USA and a Ph.D. Research Scientist in biomedical engineering also at Duke. His translational and biomedical engineering research focuses on studying cardiovascular function and end-stage heart failure. He has experience in 3-D printing, novel catheter design, tissue engineered blood vessels, and ex-vivo modification of donor hearts to improve cardiac transplantation outcomes. His clinical research is focused on clinical outcomes after adult cardiac surgery with a focus on end-stage surgical heart failure and transplantation. He is a serial inventor and entrepreneur. Contact him at muath.bishawi@duke.edu.

Simbarashe Chidyagwai is currently working on the Ph.D. degree in biomedical engineering with Duke University, Durham, NC, USA. His research involves computational fluid dynamics modeling of blood flow in congenital heart diseases. He received the B.S.E. degree in mechanical engineering from Michigan State University, East Lansing, MI, USA. Contact him at simbarashe.chidyagwai@duke.edu.

Peter Balogh received the Ph.D. degree in mechanical engineering from Rutgers University in 2018. He developed a method for modeling blood cells flowing through complex capillary networks in 3D, for which he was awarded the Acrivos Dissertation Award in Fluid Dynamics from the American Physical Society. He is currently a postdoctoral associate in the Department of Biomedical Engineering at Duke University. His research interests include computational fluid dynamics modeling of biological flows, numerical methods for complex fluid-structure interfaces, and code development for high performance computing. His current research is focused on modeling cancer cells and their transport through the circulatory system, and on investigating the remodeling of microvasculatures through comparisons with experiments. Contact him at peter.balogh@duke.edu.

Amanda Randles is currently the Alfred Winborne Mordecai and Victoria Stover Mordecai Assistant Professor of biomedical engineering with Duke University, Durham, NC, USA. She was a Lawrence Fellow with the Lawrence Livermore National Laboratory from 2013 to 2015. Prior to graduate school, she worked with IBM on the Blue Gene supercomputer. She has been the recipient of the ACM Grace Murray Hopper Award, the LLNL Lawrence Fellowship, the NIH Early Independence Award, and named as a 2017 MIT TR35 Visionary. She is a Senior Member of the National Academy of Inventors. Her research in biomedical simulation and high-performance computing focuses on the development of new computational tools that she uses to provide insight into the localization and development of human disease. She has contributed more than 40 peer-reviewed papers, more than 120 granted US patents, and has more than 100 pending patent applications. She received the bachelor's degree in physics and computer science from Duke University, and the master's degree in computer science and the Ph.D. degree in applied physics and from Harvard University, Cambridge, MA, USA. Contact her at amanda.randles@duke.edu.

Theme Article: Computational Science in the Battle Against COVID-19

Visual Analytics for Decision-Making During Pandemics

Audrey Reinert
University of Oklahoma

Luke S. Snyder
Purdue University

Jieqiong Zhao
Purdue University

Andrew S. Fox
University of Oklahoma

Dean F. Hougen
University of Oklahoma

Charles Nicholson
University of Oklahoma

David S. Ebert
University of Oklahoma

***Abstract*—We introduce a trans-disciplinary collaboration between researchers, healthcare practitioners, and community health partners in the Southwestern U.S. to enable improved management, response, and recovery to our current pandemic and for future health emergencies. Our Center work enables effective and efficient decision-making through interactive, human-guided analytical environments. We discuss our PanViz 2.0 system, a visual analytics application for supporting pandemic preparedness through a tightly coupled epidemiological model and interactive interface. We discuss our framework, current work, and plans to extend the system with exploration of what-if scenarios, interactive machine learning for model parameter inference, and analysis of mitigation strategies to facilitate decision-making during public health crises.**

Making timely, effective, science-based decisions to mitigate the impact of a pandemic is a very difficult and highly complex task requiring a decision-maker to evaluate multiple disparate data sources. Decisions such as when to reopen require collecting and integrating accurate information on an array of the characteristics, including: disease spread dynamics, prevalence of infections when initially detected, capacity and supplies available at all hospitals in the state, the

Digital Object Identifier 10.1109/MCSE.2020.3023288
Date of publication 14 September 2020; date of current version 9 October 2020.

effectiveness of each potential mitigation strategy, the delay in adopting mitigation behaviors, and assessing public compliance with public health measures, and other public behavior.

Our approach to addressing these problems is to combine available data from State Departments of Public Health, population demographics, fine-scale population behavior and mobility data and predictions, multiple advanced epidemiological models, epidemiologists, and continuously refined predictions and data-driven model parameters, into a decision-making and situational awareness dashboard to enable evaluation of current situation and strategies for implementing nonpharmaceutical interventions (NPIs).

Given that the data landscape surrounding COVID-19 is evolving so quickly, we emphasize that work will continue for quite some time. We present the initial results of a trans-disciplinary collaboration between researchers, healthcare practitioners, and community health partners in the Southwestern U.S. to help enable improved management, response, and recovery options for our current pandemic and future health emergencies through the development of an integrated data dashboard. This dashboard and associated technical advances enable a decision maker to:

- model and visualize how NPIs impact the spread of COVID-19;
- monitor the spread of COVID-19 related news on various social media platforms;
- design effective risk communication strategies to ensure compliance with NPIs.

These advances are central to the new **C**enter for **I**ntegrated **P**ublic **H**ealth **M**onitoring, **A**nalysis and **D**ecision-making (CIPH-MAD), a collaboration of researchers and decision-makers dedicated to predicting and mitigating public health emergencies in the South-western US. This work is applicable to future pandemic and public health emergencies by providing a framework to integrate and synthesize multiple disparate data sources.

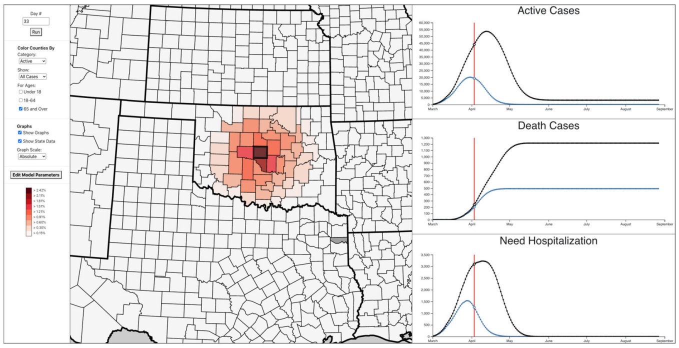
BACKGROUND

Early detection and action are key to mitigating the effects, as demonstrated with the successful response to Ebola. Effective detection, mitigation, and response rely on accurate information, analytics, and predictions of the effect of interdiction/mitigation strategies. Over the past nine months, there has been great progress in gathering data on the COVID-19 pandemic. New data sources, including social mobility data and public health electronic surveillance data, are becoming available and are being used effectively in a few locations. These new sources greatly increase situational awareness and provide rapid feedback regarding the effectiveness of various public health actions and policies.[1] However, the available data are often conflicting, biased due to sampling, and incomplete. Given the size, scope, and complexity of pandemic data, it can be difficult for a decision-maker to gauge the effectiveness of different mitigation strategies without effective computational support.

The goal is to create an interactive decision-support and dashboard system for interactive public health situational awareness, planning, and response.

Fortunately, *Visual Analytics* (VA) tools and techniques allow a decision-maker to address certain problems whose inherent size, complexity, and need for closely coupled human and machine analysis may make them otherwise intractable. The advantage of using a VA system in a disease modeling and mitigation context is that a decision-maker can compare and contrast the effect different intervention techniques—including social distancing, mask usage, or closing schools—will have on the spread of a given disease throughout a region. Furthermore, a decision-maker can use a VA system to explain to other stakeholders and the public what the near- and long-term effects of a particular decision will be. VA systems can communicate complicated and nuanced findings from statistical models to a larger audience.

Previous VA work has demonstrated the effectiveness of interactive decision support tools at identifying intervention policies. One tool, PanViz,[2] was initially developed to provide public health officials from the Indiana State Department of Health with a suite of visual analytic tools for analyzing pandemic influenza spread, while enabling these officials to analyze various decision points (e.g., school closure, strategic national stockpile release) and their impact on disease spread. The tool allowed for


Day # 33

Color Counties By
Category: Active
Show: All Cases
For Ages:
☐ Under 18
☐ 18-64
☑ 65 and Over

Graphs
☑ Show Graphs
☑ Show State Data
Graph Scale: Absolute

Figure 1. Overview of new PanViz 2.0: (left) control panel for fine-tune of model parameters and configuration of visual representations, (middle) spatial distribution of a selected case category, (right) temporal visualization of different case categories.

demographic filtering by age ranges, and interactive manipulation of model parameters to allow users to create various levels of pandemic severity in order to assess various situations. Using this tool, health officials could analyze resources and decision outcomes in order to prepare more effective measures for potential pandemics. This work was expanded upon in subsequent iterations to model the spread of Rift Valley Fever and Dengue Fever and predict successful public health containment procedures.[3,4] PanViz is not the only VA tool developed for tracking and predicting the spread of pandemics; however, it is one of the only ones that focuses on multiple levels of preparedness and mitigation.

A limitation of PanViz is that it was not initially designed to update with incoming surveillance data. Moreover, it does not accommodate behavioral or social science data. Understanding behavioral data is beneficial to an analyst because it permits a deeper understanding of how the public views, interprets, and responds to information about the pandemic, enabling a decision-maker to tailor public announcements about the need to close schools or engage in social distancing while gauging how the public perceives the burden. To address this gap, members of our team are currently engaged in a continuously updated behavioral monitoring survey, focused on COVID-19 related beliefs and behaviors in the U.S. (NSF RAPID Grant 2026763), as well as analysis of Twitter data related to COVID-19. We are also collaborating with researchers at Purdue University, University of West Florida, and Arizona State University to explore social mobility data for more effective measures of the impact of NPIs (NSF RAPID grant 2027524). These data will be leveraged to permit analysis of how changing perceptions and behaviors among the public affect later infection rates across regions of the US. The results will be incorporated into PanViz 2.0, allowing for a blending of more traditional **S**usceptible, **I**nfected and **R**ecovered (SIR) infection spread models, data-driven models, and agent-based models with up-to-date perception, attitudinal, and effectiveness models.

PANVIZ 2.0 OVERVIEW

The decision making in the context of combating COVID-19 or similar future pandemics requires a workflow to transform raw data into actionable information including statistics, visualization, and interaction from a variety of sources. The workflow employed by PanViz2.0 was inspired in part by our previous research developing and evaluating novel VA applications.[5] Below, we introduce the reader to the PanViz 2.0 interface and computational architecture while providing a road map for planned improvements.

PanViz 2.0 Design

Our current decision-support framework system builds upon our earlier PanViz work in public health syndromic surveillance, pandemic preparedness, and decision support for other person-spread and mosquito-spread conditions.[2] PanViz was used extensively during the 2008–2012 pandemic preparedness activities in the United States by numerous counties and states. The PanViz 2.0 visual analytic framework prototype is a re-engineered design based on our experience in developing and deploying visual analytic decision support systems over the past decade. The prototype, shown in Figure 1, is built upon a mathematical epidemic model to calculate population dynamics and infection rate data and enables decision-makers to interactively choose mitigation strategies and see the impact of their decisions.[6] PanVis 2.0 improves upon PanViz in several respects, including the system architecture, ability to incorporate behavioral, social movement and dynamics, and observed data for improving accuracy and predictions in the system, and adds interactive decision-making features. In concert, these changes facilitate data- and human-guided, science-based AI-driven analysis of public health data for improved preparedness.

PanViz 2.0 System Architecture and Interface

PanViz 2.0 has been converted from a desktop system to an intuitive web-based application to address the portability and scalability problems, as multiple users can now access the system from any web browser. It consists of a backend Flask server and frontend web user interface. The server dynamically executes the model simulation for a given set of parameters (as controlled and updated by the user). The web interface (see Figure 1) uses the React JavaScript library for efficient frontend rendering and interactivity. Users can modify the model features, parameters, and parameters to be visualized (first panel on left-hand side of Figure 1), interactively visualize the model simulation over time in the geographic space (center panel of Figure 1), and simultaneously visualize time series data of county-, state-, and national-level infection, death, and hospitalization numbers (panels on right-hand side of Figure 1).

PanViz 2.0 also supports county-level disease parameterization. Users can configure model parameters for each county to appropriately account for locally dependent transmission dynamics, such as the demographic impact, spread rate (as controlled by population density), mortality rate, implemented decision measures, and hospital capacity. We plan to incorporate sophisticated machine learning and data mining techniques to learn highly accurate county-level parameters from collected data for improved decision-making.

Base Epidemiological Model

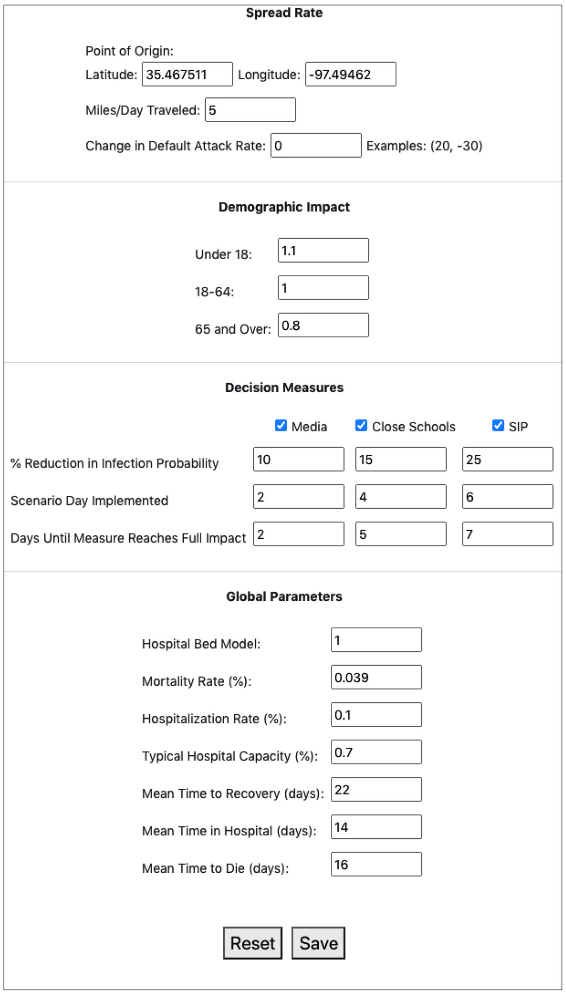
The mathematical model underpinning PanViz (Malone *et al.*[6]) calculates disease dynamics per county using a system of nonlinear difference equations derived from traditional epidemic models with homogeneous population mixing derived from previous influenza and pandemic data and integrates an airport transportation spread model.

Disease dynamics are evaluated by combining user-supplied values for county demographics and population density, mortality and recovery rate of the disease, hospitalization rate, and baseline and modified disease prevalence. The baseline prevalence is approximated using the gross attack rate, which is the percentage of the entire U.S. population that will have the disease if no interventions are enacted. The total number of infections in a county is calculated by age group and is the product of county population, age group specific disease modifiers, and the presence of decision measures. Simulated individuals can be either healthy, infected, recovered, or deceased and will transition between these states at a certain rate. The likelihood an individual from any age category will become infected is influenced by the population density of the county they reside in and the baseline and modified prevalence of the disease.[6] For more information on the formulation of the model, please see the paper by Maciejewski *et al.*[3,] and Malone *et al.*[6]

The model assumes a disease originates from a user-defined location. The county to county spread rate is dependent on the distance from the origin to the county centroid, the county population density, and demographic composition. The inclusion of the airport transportation dynamics enables the transmission within a day of the disease to all connected airport hubs once the disease reaches an airport.

Time-Based Interdiction for Interactive Decision-Making

Decision measures are critical for impeding virus spread, although such measures may vary considerably over time and space. Comparative analysis of such decision measures is equally crucial for assessing how effective and suitable they may be for different situations. In particular, it is important to answer policy questions such as "*What decision measures should be implemented?*" PanViz 2.0 will allow users to answer such questions by visually comparing the effect that different decision measures have on virus spread, as well as infection, mortality, and hospitalization rates (see Figures 2 and 3). When the exact parameters of such interdiction strategies are difficult to estimate *a priori* or from previous pandemic data due to transmission novelty, users will be able to explore different estimates, such as in Figure 4, and update them as new data are collected.



Spread Rate

Point of Origin:
Latitude: 35.467511 Longitude: -97.49462

Miles/Day Traveled: 5

Change in Default Attack Rate: 0 Examples: (20, -30)

Demographic Impact

Under 18:	1.1
18-64:	1
65 and Over:	0.8

Decision Measures

	✓ Media	✓ Close Schools	✓ SIP
% Reduction in Infection Probability	10	15	25
Scenario Day Implemented	2	4	6
Days Until Measure Reaches Full Impact	2	5	7

Global Parameters

Hospital Bed Model:	1
Mortality Rate (%):	0.039
Hospitalization Rate (%):	0.1
Typical Hospital Capacity (%):	0.7
Mean Time to Recovery (days):	22
Mean Time in Hospital (days):	14
Mean Time to Die (days):	16

Figure 2. Mockup of the next generation of the PanViz 2.0 UI. Users will be able to visualize the impacts of different decision measures and what-if scenarios on infection, death, and hospitalization counts, compare epidemiological model data with observed data, and estimate future predictions from observed data with interactive machine learning. These views will be tightly linked such that users can input different interdiction strategies and parameter combinations and visualize the resulting model data in other views.

Public officials may be interested in determining when and how long citizens should be required to wear masks, practice social distancing, and avoid public gatherings to effectively curtail the spread of COVID-19. Officials can investigate their concerns with PanViz 2.0 by comparing the virus spread and number of infections when these intervention measures are implemented early on in the pandemic as opposed to later. Officials can also assess these measures' effectiveness under environmental assumptions such as the probability of an individual complying with mask wearing or the continued occurrence of large social gatherings. Using such results, officials can appropriately determine the best course of action for intervention policies. The system will also enable the starting and stopping of different measures over time during each wave of virus spread.

Incorporating Epidemiological Expertise

We have been working with state and local public health officials to integrate their feedback, domain expertise, and perspectives into the design and implementation of PanViz 2.0. Since portions of the team have worked with public health professionals in the past, we recognize the need to closely collaborate with and include domain expert opinions to ensure the PanViz 2.0 interface and application components support effective and actionable decision-making.

Planned Work: PanViz 2.0 for COVID-19

Due to the novelty of COVID-19, traditional epidemiological models may fail to accurately simulate the virus spread. Current epidemiology studies suggest that the virus spreads from person to person just as any other virus spreads, but with potentially different parameters for when symptoms start, time when contagious, etc. This problem is exacerbated for lower population regions and counties with limited case histories or areas with unique characteristics that small towns with large university student populations. At the same time, these areas are also the least prepared for an onslaught of COVID-19 cases.[7,8]

Regional hospitals serving such areas need to evaluate how best to utilize their limited budgets and resources, to meet the upcoming demand; however, access to good information to support such decisions is poor. Therefore, PanViz 2.0 will ingest collected data for data-driven model comparisons against baseline simulations. In particular, users will be able to visualize the differences between baseline simulations and observed COVID-19 data to explore and infer model parameters and adjust the model's settings for future predictions (see Figure 2). PanViz 2.0 will also incorporate interactive machine

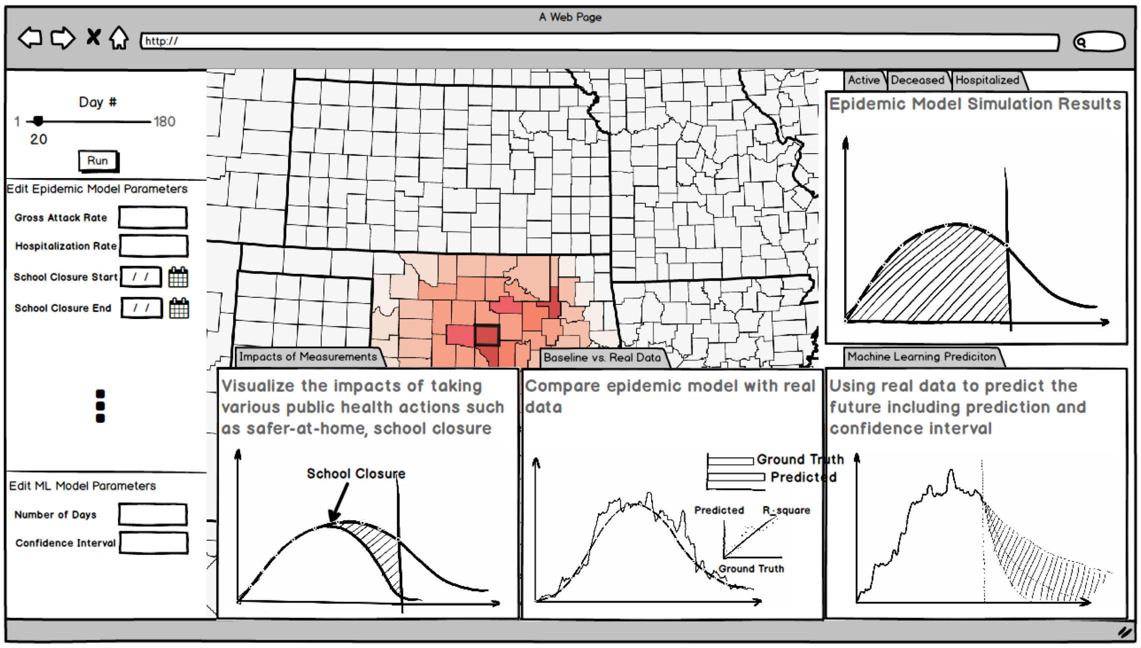

A Web Page
http://
Day #
1
180
20
Run
Edit Epidemic Model Parameters
Gross Attack Rate
Hospitalization Rate
School Closure Start
School Closure End
Edit ML Model Parameters
Number of Days
Confidence Interval
Active
Deceased
Hospitalized
Epidemic Model Simulation Results
Impacts of Measurements
Visualize the impacts of taking various public health actions such as safer-at-home, school closure
School Closure
Baseline vs. Real Data
Compare epidemic model with real data
Ground Truth
Predicted
Predicted
R-square
Ground Truth
Machine Learning Prediciton
Using real data to predict the future including prediction and confidence interval

Figure 3. Users can compare the cumulative effects (here in terms of lost and saved lives) of various decision measures against the baseline simulation to assess their effectiveness. Example result shown is from previous work with Rift Valley Fever.[4]

learning for user-steerable parameter learning to improve model predictions. Multiple model simulations with different parameters will be used to train a neural network to predict the corresponding parameters after human-review for appropriateness. The trained model can then be used to predict parameters for real-life data that can be further tuned and adapted through interactive user analysis. Our previous work in syndromic surveillance to more accurately analyze surveillance data, reducing anomaly false positives while modeling and predicting incident occurrence in the upcoming 14 days,[3] will be incorporated as well to harness trustable data-driven predictions for interdiction planning.

While past data can be useful for tuning the model, there is still a degree of uncertainty regarding future trends. Therefore, PanViz 2.0 will also support sophisticated exploration of what-if scenarios by allowing users to select different parameter combinations and visually compare them (shown in Figure 2). For example, the user may be interested in determining how the virus spread changes under different spread rates, hospital capacities, and demographic impacts. Finally, we are in the implementation and refining phase of both the visual design and technical implementation of PanViz 2.0.

PANVIZ 2.0 AND SOCIAL MEDIA DATA

Since early 2020, part of our team at the University of Oklahoma National Institute for Risk and Resilience (NIRR) has pursued several projects focused on the COVID-19 pandemic. In January, they implemented a broad collection of social media posts from Twitter's API using a basket of search terms. That collection includes approximately 300 million posts, amounting to half a TB of data. These data are used to identify the evolving array of COVID-19 communication

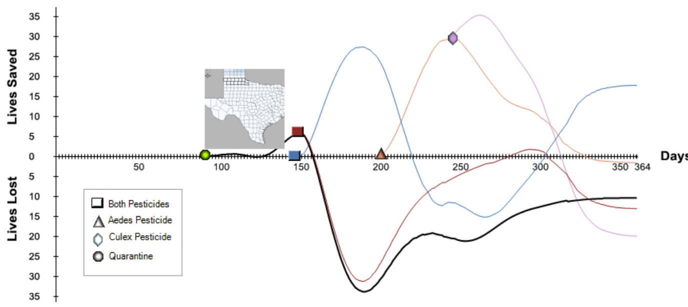


Figure 4. User-configurable parameter window. Users can set the appropriate parameters for each county.

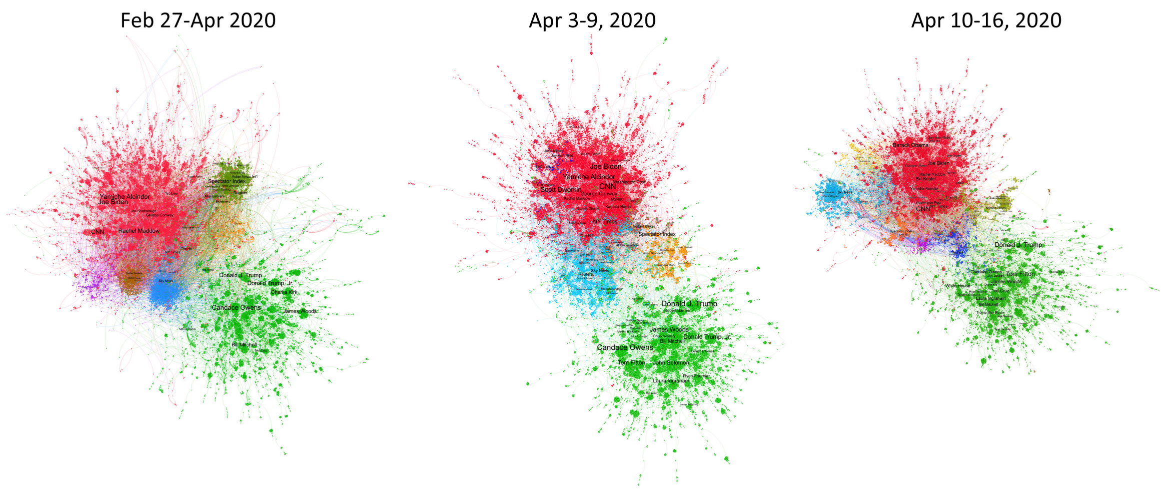


Figure 5. Depicts the networks of Twitter users posting about COVID-19 over the period of February 27th through April 16th, 2020. Clusters were identified using the Louvain community detection algorithm.

networks using the Louvain community detection algorithm on the largest connected component of the retweet network drawn from the most prolific accounts (minimum three tweets per day) to prune the network. Weekly samples are drawn from the top tweets (according to PageRank scores) within each cluster and categorized by human coders to track the content of social narratives.

As shown in Figure 5, the constellation of network clusters has been quite stable, with seven or eight groupings consisting of relatively consistent leading members. The resulting weekly network snapshots reveal a polarized network structure that reflects current divisions in American politics. Interestingly, the political right displays a remarkable level of stability while the left has exhibited more dynamic cluster with the larger moderate left gradually incorporating a smaller progressive group over time. A common characteristic of these network communities is the presence of dense clusters of users around popular politicians and media figures with relatively short communication pathways that enable the rapid transmission of information.

Of particular interest are patterns of misinformation about the nature, transmission, effects, and protective actions associated with COVID-19 within each of the more stable network clusters. Preliminary analysis of the network structure and content of the most prominent accounts and tweets has found a number of coherent misinformation narratives, including conspiracy theories, cures, and other statements about COVID-19 that are based on verifiably false claims, that have spread throughout the social media landscape. These narratives include claims that COVID-19 is the product of a shadowy conspiracy of powerful individuals, the virus originated as a bioweapon, COVID-19 is no worse than seasonal flu, and hydroxychloroquine is an effective cure. Among the false narratives identified, a disproportionate number of these misleading claims regularly have appeared within the conservative right community on social media, but it remains to be seen if this flow of misinformation will shift.

Starting in mid-March, a rolling nationwide survey, with weekly representative subsamples, was implemented to track the patterns of awareness of and belief about the COVID-19 misinformation identified in the Twitter social media collection. The survey permits assessment of the ways in which social media misinformation (and efforts to counter that misinformation) affect evolving public concern about and response to the pandemic. A key interest is in understanding changes over time in public trust for experts and individual willingness to engage in protective actions.

Also in March, Governor Kevin Stitt of Oklahoma asked the OU NIRR, as part of a team of modelers, to provide regular updates on the projected spread of COVID-19 related hospitalizations and ICU demand within Oklahoma. Early in the pandemic, models were showing wide discrepancies in these estimates,[9] with underlying uncertainties clouding urgent policy decisions. The NIRR proposed and implemented a modeling ensemble approach, utilizing a range of nationally recognized models to provide the Governor, the State Epidemiologist, and cabinet with the range of regularly updated projections for the state. This effort extended through May 2020, when the pandemic (appeared to have) peaked in Oklahoma. These ensemble models are being integrated into PanViz 2.0.

PANVIZ 2.0 AND MOBILITY DATA

One question implied by the current work is: To answer this question, in addition to our work in surveying, social media data collection, and smart app user-provided data, we and our collaborators repurposed existing cyberinfrastructures to analyze the risk of future epidemics in crowded locations by using real-time webcam videos and data provided by location-based services (LBS) (NSF RAPID grant 2027524). The proposed solution incorporates pedestrian dynamics to assess if individuals are complying with social distancing.

How does a decision-maker assess if individuals are complying with public health measures such as social distancing?

We believe this research will yield actionable data that can be incorporated into PanViz 2.0 to asses compliance with NPIs, since LBS data can be used to identify crowded locations with a spatial resolution in the tens of meters, and video data can assess how individuals congregate within and move through large public spaces.

From a decision-making perspective, there are several benefits of this data. The first is the ability to identify potential transmission hotspots on a very-fine grain level, allowing a decision-maker to assess if a particular store or nursing home is at risk of becoming a hotspot. Second, the collection and analysis of LBS and video data can be used in contract tracing applications. Looking toward the future, this information can be used to redesign public spaces for pandemic safety.

These data sources, along with social media data and user-provided app data, have been integrated with the PanViz 2.0 architecture to provide a decision-maker with a series of scaling intervention options. A decision-maker can now suggest a series of behavioral "nudges" be issued to individuals in high-risk areas to remind them to comply with public health practices or simply alter them of the risk. On a regional level, the decision-maker could assess public perception of and compliance with regional level public health measures. Finally, at a national level, a decision-maker can surmise how the public at large views public health measures and asses the types of locations at risk of becoming a hotspot.

PANVIZ 2.0 AND CO-ADVISOR

Gathering detailed information at the individual level and providing individualized communications to help ordinary people respond are also goals of our Center. Part of our team, based in OU's School of Computer Science, has prototyped the Co-Advisor application for this purpose. Co-Advisor is a smart-device app designed to ingest data on the activities of users and others, as well as individual health data, in order to provide users with current and predicted risk assessments based on their current and planned behaviors. The app is designed to communicate risk assessments in a clear, simple, and timely manner. Co-Advisor ingests data in three categories: user activities, user health data, and activities of others. The types of data collected about individual activities include mobility, records of personal behaviors such as mask wearing, and social behaviors such as work environment. An individual user can supply records of their symptoms and/or the system can automatically detect them. Finally, Co-Advisor ingests data about the behaviors of others that users have come in contact with, such as whether a visited location was crowded. These data are collected by sensors including Bluetooth and GPS tracking, and user inputs.[10] One of the primary functions of Co-Advisor is to convert the collected data into actionable information. An example of this can be viewed as a user story. If a user wants to engage in contract tracing, the user enables location sharing (while preserving privacy and security[11]) to help assess where they had been during the past 20 days. This information allows Co-Advisor to determine if they

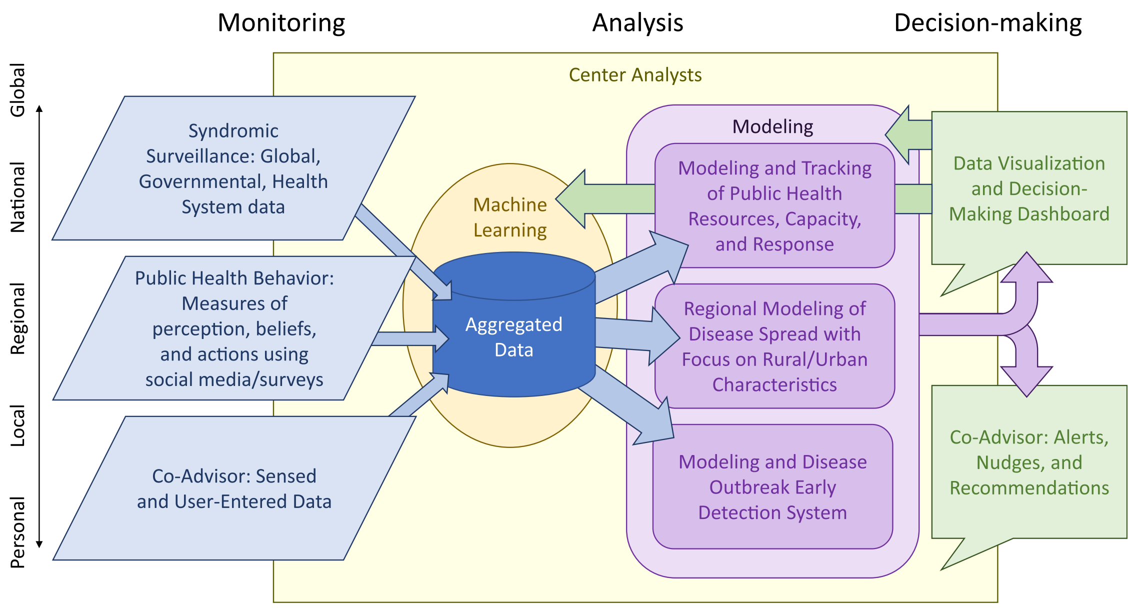

Monitoring
Analysis
Decision-making
Global
National
Regional
Local
Personal
Center Analysts
Syndromic Surveillance: Global, Governmental, Health System data
Public Health Behavior: Measures of perception, beliefs, and actions using social media/surveys
Co-Advisor: Sensed and User-Entered Data
Machine Learning
Aggregated Data
Modeling
Modeling and Tracking of Public Health Resources, Capacity, and Response
Regional Modeling of Disease Spread with Focus on Rural/Urban Characteristics
Modeling and Disease Outbreak Early Detection System
Data Visualization and Decision-Making Dashboard
Co-Advisor: Alerts, Nudges, and Recommendations

Figure 6. Configuration of CIPH-MAD, which facilitates public health emergency planning and response through integrated data sources (monitoring), advanced modeling (analysis), and data visualization and risk alert communications (decision-making).

had come in contact with others who were likely to have COVID-19. Assuming the user had been in substantial contact with an infected individual, Co-Advisor informs them of that risk. Effective risk communication is clear, simple, timely, and incorporates awareness of the cultural context of the user. Co-Advisor uses interactive communication strategies in lay language with supporting evidence to make biomedical prevention messages credible in affected communities. This risk is communicated via alerts triggered by public or private data; nudges, which encourage individuals to change their behavior and can be used as just-in-time adaptive interventions;[12] and detailed recommendations based on user behavior.

PLAN FOR A CENTER CIPH-MAD

As mentioned in the introduction, we plan to form a Southwestern U.S. collaboration to enable more effective surveillance, planning, mitigation, and response to public health emergencies. To state, we have initiated a **C**enter for **I**ntegrated **P**ublic **H**ealth **M**onitoring, **A**nalysis and **D**ecision-making (CIPH-MAD). This Center will pursue opportunities for generating and harnessing new data sources to improve planning, detection, response, communication, and management for pandemics like COVID-19 (see Figure 6). Currently, there is no deployed system that integrates these data and capabilities into a unified decision-making system for hospital and government decision-makers and includes mitigation strategy planning. While large, population dense areas see more infectious disease cases, they also have greater resources. Oklahoma's budget per capita is only 52.6% of New York's. Accurate disease projections are potentially "mission critical" for Oklahoma and other Southwestern states given this relative shortfall of resources. Moreover, Oklahoma and many other states lack real-time electronic syndromic surveillance to provide the base data needed for accurate situational surveillance, virus spread status, and measurement of mitigation actions.

Included in the Center will be development and deployment of a "Co-Advisor" app that would build on the functionality the Google/Apple social-distancing/contact-tracing capabilities to provide user advice and information while providing anonymized input to CIPH-MAD's integrated public health planning system, built upon PanViz 2.0. These data, as well as the social media and survey data described above, will be utilized to expand the capability and functionality of existing pandemic models (like already developed models in PanViz). Furthermore, we will create the PanViz 2.0 decision support framework that is sensitive to the community-specific characteristics and/or specialized subpopulations. We will accomplish this in part through machine learning to create datasets amenable to statistical analyses for forecasting, incorporating features such as population density, income dispersion, age distribution, mobility data, and unique factors such as Native American population and presence of higher education facilities. Small area estimation techniques including missing data imputation for survey data and Bayesian estimation will enhance the informational value of datasets and improve forecasting accuracy. The overall focus is on capturing the synergies across the computing, public health and social sciences to build a data-driven, integrated healthcare modeling system to both act as a sentry for emerging pandemics and as a tool for managing them.

In the coming months, we will expand CIPH-MAD to include researchers from Arizona State University and the University of Texas, Austin. This will expand the reach of the center to cover the South Western portion of the United States.

CONCLUSION

Making timely, effective, science-based decisions to mitigate the impact of a pandemic is a very difficult and highly complex task. As recent news events have shown, the pressures facing public health officials are immense and life-altering for thousands of individuals. Even a decision as seemingly small as when to announce a particular policy can save the lives of tens of thousands of individuals.

However, making these complicated decisions without computational support can be very difficult. In this article, we presented the initial results of a trans-disciplinary collaboration between researchers, healthcare practitioners, and community health partners in the Southwestern U.S. to help enable improved management, response, and recovery to our current pandemic and for future health emergencies through the development of an integrated data dashboard. Our aim is to provide decision-makers with a tool to help them synthesize and visualize a wide variety of data types—ranging from hospital capacity to Facebook posts—to help them evaluate the outcomes of various decisions to further their goal of making life-saving choices. Given that the data landscape surrounding COVID-19 are evolving so fast and the little understood nature of the disease, we emphasize that work will continue for quite some time. Our work to date does facilitate an understanding of the consequences of various interventions. Furthermore, our work is imminently applicable to future pandemic and public health emergencies by providing a framework to integrate and synthesize multiple disparate data sources.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under Grant SES-2026763 and Grant 2027524. The authors would like to thank Dr. H. Jenkins-Smith for his contributions.

REFERENCES

1. S. J. Grannis, K. Stevens, and R. Merriwether, "Leveraging health information exchange to support public health situational awareness: The Indiana experience," *Online J. Public Health Informat.*, vol. 2, no. 2, Oct. 2010, doi: ojphi.v2i2.3213.
2. R. Maciejewski *et al.*, "A pandemic influenza modeling and visualization tool," *J. Vis. Lang. Comput.*, vol. 22, no. 4, pp. 268–278, 2011.
3. R. Maciejewski *et al.*, "Enabling syndromic surveillance in Pakistan," *Online J. Public Health Informat.*, vol. 5, no. 1, Mar. 2013. doi: 10.5210/ojphi.v5i1.4393.
4. S. Afzal, R. Maciejewski, and D. S. Ebert, "Visual analytics decision support environment for epidemic modeling and response evaluation," in *Proc. IEEE Symp. Vis. Analytics Sci. Technol.*, Dec. 2011, pp. 191–200.
5. M. Chen and D. S. Ebert, "An ontological framework for supporting the design and evaluation of visual analytics systems," *Comput. Graph. Forum*, vol. 38, no. 3, pp. 131–144, 2019.
6. J. D. Malone *et al.*, "US airport entry screening in response to pandemic influenza: Modeling and analysis," *Travel Med. Infectious Disease*, vol. 7, no. 4, pp. 181–191, 2009.

7. A. Roy, J. Jose, A. Sunil, N. Gautam, D. Nathalia, and A. Suresh, “Prediction and spread visualization of COVID-19 pandemic using machine learning,” *Preprints*, 2020, doi: 10.20944/preprints202005.0147.v1.
8. S. K. Dey, M. M. Rahman, U. R. Siddiqi, and A. Howlader, “Analyzing the epidemiological outbreak of COVID-19: A visual exploratory data analysis approach,” *J. Med. Virol.*, vol. 92, no. 6, pp. 632–638, 2020.
9. Q. Bui, J. Katz, A. Parlapiano, and M. Sanger-Katz, “ What 5 Coronavirus models say the next month will look like,” *The New York Times*, Apr. 2020, Accessed: Jul. 2, 2020. [Online]. Available: https://www.nytimes.com/interactive/2020/04/22/upshot/coronavirus-models.html
10. M. Wirz, T. Franke, D. Roggen, E. Mitleton-Kelly, P. Lukowicz, and G. Tröster, “Probing crowd density through smartphones in city-scale mass gatherings,” *EPJ Data Sci.*, vol. 2, no. 5, Dec. 2013, [Online]. Available: http://www.epjdatascience.com/content/2/1/5, doi: 10.1140/epjds17.
11. L. Baumgärtner, A. Dmitrienko, B. Freisleben, and A. Gruler, “Mind the GAP: Security & privacy risks of contact tracing apps,” Jun. 2020. [Online]. Available: https://arxiv.org/abs/2006.05914
12. L. Wang and L. C. Miller, “Just-in-the-moment adaptive interventions (JITAI): A meta-analytical review,” *Health Commun.*, Routledge, vol. 35, no. 12, pp. 1–14, Sep. 2019. doi: 10.1080/10410236.2019.1652388.

Audrey Reinert is currently a Postdoctoral Researcher with the University of Oklahoma’s Data Science Institute for Societal Challenges (DISC Center), Norman, OK, USA. Her research interests lie at the intersection of human centered computing, data visualization, and human–machine interaction. She received the bachelor’s degree in cognitive neuropsychology from the University of California, San Diego, San Diego, CA, USA, in 2012, the master’s degree in human–computer interaction from the Georgia Institute of Technology, Atlanta, GA, USA, in 2015, and the Ph.D. degree in industrial engineering from Purdue University, West Lafayette, IN, USA, in 2019. She is a member of IEEE, HFES, and the AGU. Contact her at areinert@ou.edu.

Luke S. Snyder is currently a Research Associate with the University of Oklahoma, Norman, OK, USA. His research interests include visual analytics, visualization, interactive machine learning, and human–computer interaction. He received the B.S. degree in computer science from the University of Arkansas, Fayetteville, AR, USA, in 2018, and the M.S. degree in computer science from Purdue University, West Lafayette, IN, USA, in 2020. He is a member of IEEE. Contact him at snyder@ou.edu.

Jieqiong Zhao received the master’s degree in computer science from Tufts University, Medford, MA, USA, in 2013 and the Ph.D. degree in computer engineering from Purdue University, West Lafayette, IN, USA, in 2020. Her broad research interests include visual analytics, information visualization, human–computer interaction, and applied artificial intelligence and machine learning. Contact her at zhao413@purdue.edu.

Andrew S. Fox is currently a Research Scientist with the Center for Risk and Crisis Management, part of the University of Oklahoma’s National Institute for Risk and Resilience, Norman, OK, USA. His research interests include public policy, risk, and social networks. He received the Ph.D. degree in political science, University of Oklahoma, in 2019. He is a member of American Political Science Association and Society for Risk Analysis. Contact him at asfox@ou.edu.

Dean F. Hougen is currently an Associate Director and Associate Professor with the School of Computer Science, Gallogly College of Engineering, University of Oklahoma, Norman, OK, USA, where he leads the Robotics, Evolution, Adaptation, and Learning Laboratory (REAL Lab). His research interests span the field of artificial intelligence, with emphases in robotics, machine learning, and multiagent systems. He received the bachelor’s of Science in computer science from Iowa State University, Ames, IA, USA, with minors in mathematics and philosophy, a graduate minor in cognitive science, and Doctor of Philosophy in computer science from the University of Minnesota, Minneapolis, MN, USA. He is a member of the IEEE as well as AAAI, ACM, ASEE, INNS, and ISAL. Contact him at hougen@ou.edu.

Charles Nicholson focuses his research on development and application of statistical and machine learning algorithms that improve the speed, efficiency, and quality of insight from analytics on large data systems. He received the undergraduate degrees in mathematics and physics, the master's of science degree in decision technology from the University of North Texas, Denton, TX, USA, and the doctorate in operations research from Southern Methodist University, Dallas, TX, USA, specializing in optimization of network flow problems. Contact him at cnicholson@ou.edu.

David S. Ebert is currently an Associate Vice President for Research and Partnerships, the Gallogly Chair Professor of electrical and computer engineering, and the Director of the Data Institute for Societal Challenges, University of Oklahoma, Norman, OK, USA. He is the recipient of the 2017 IEEE Computer Society vgTC Technical Achievement Award, member of the IEEE vgTC Visualization Academy, an adjunct Professor of electrical and computer engineering with Purdue University, West Lafayette, IN, USA, and the Director of the Visual Analytics for Command Control and Interoperability Center (VACCINE), the Visualization Science team of the Department of Homeland Security's Visual Analytics and Data Analytics Emeritus Center of Excellence. He received the Ph.D. degree in computer and information science from The Ohio State University, Columbus, OH, USA and performs research in visual analytics, novel visualization techniques, interactive machine learning and explainable AI, human–computer teaming, advanced predictive analytics, and procedural abstraction of complex, massive data. He is an IEEE Fellow. Contact him at ebert@ou.edu.

IEEE Computer Society Volunteer Service Awards

Nominations accepted throughout the year.

T. Michael Elliott Distinguished Service Certificate

Highest service award in recognition for distinguished service to the IEEE Computer Society at a level of dedication rarely demonstrated. i.e., initiating a Society program or conference, continuing officership, or long-term and active service on Society committees.

Meritorious Service Certificate

Second highest level service certificate for meritorious service to an IEEE Computer Society-sponsored activity. i.e., significant as an editorship, committee, Computer Society officer, or conference general or program chair.

Outstanding Contribution Certificate

Third highest level service certificate for a specific achievement of major value to the IEEE Computer Society, i.e., launching a major conference series, a specific publication, standards and model curricula.

Continuous Service Certificate

Recognize and encourage ongoing involvement of volunteers in IEEE Computer Society programs. The initial certificate may be awarded after three years of continuous service.

Certificate of Appreciation

Areas of contribution would include service with a conference organizing or program committee. May be given to subcommittee members in lieu of a letter of appreciation.

IEEE COMPUTER SOCIETY

IEEE

Nominations

Submit your nomination at
http://bit.ly/computersocietyawards

Contact us at
awards@computer.org

Optimal Kernel Design for Finite-Element Numerical Integration on GPUs

Krzysztof Banaś
AGH University of Science and Technology

Jan Bielański
AGH University of Science and Technology

Filip Krużel
Cracow University of Technology

***Abstract*—This article presents the design and optimization of the GPU kernels for numerical integration, as it is applied in the standard form in finite-element codes. The optimization process employs autotuning, with the main emphasis on the placement of variables in the shared memory or registers. OpenCL and the first order finite-element method (FEM) approximation are selected for code design, but the techniques are also applicable to the CUDA programming model and other types of finite-element discretizations (including discontinuous Galerkin and isogeometric). The autotuning optimization is performed for four example graphics processors and the obtained results are discussed.**

The effective exploitation of the computational power of graphics processors (GPUs) requires efforts of both theoretical and practical matters. Theoretical performance modeling can indicate the most important factors influencing the execution characteristics for considered algorithms.[1,2] The detailed quantitative results for particular GPU hardware can still be difficult to obtain and use in practice. Moreover, the results for one architecture are usually not applicable to the others, even for similar processors, e.g., forming a sequence of generations for a specific family.[3]

The autotuning techniques that form an experimental extension of theoretical models and have been applied with success for CPUs since the 1990s,[4] are becoming more and more popular for the GPU architectures.[5] The specific characteristics of the GPUs and their most popular programming models—CUDA and OpenCL—offer some fundamental optimization parameters, such as the number of threads, their organization into global and local index spaces (using the notion of threadblocks or workgroups), as well as run time and compiler optimization options, including the number of registers employed or the level of loop unrolling.[6]

Digital Object Identifier 10.1109/MCSE.2019.2940656
Date of publication 18 September 2019; date of current version 9 October 2020.

In this article, we consider several other options for GPU autotuning that require more substantial intrusion into the optimized code. The algorithm that we use for our study is the finite-element numerical integration, an important algorithm used in almost all the FEM simulation codes. It is used for creation of a system of linear equations, the solution of which gives degrees of freedom for the finite-element approximations. The work presented in this paper is a continuation of our investigations on execution of finite-element core calculations on modern hardware platforms presented, e.g., as in paper by Płaszewski *et al.* [7,8]

ALGORITHM OF FINITE-ELEMENT NUMERICAL INTEGRATION

The goal of the finite-element numerical integration algorithm is to create, for each finite element, a local matrix A^e (element stiffness matrix) that is further used in calculations in a manner specific to a particular solution strategy (to simplify and clarify the analysis, we neglect in this paper the creation of the vector of the right hand side of the system).

We omit the details of the context in which the numerical integration algorithm is used within finite-element calculations. In particular, we do not discuss the problem of assembling the element stiffness matrices into the global system matrix[9] (allowing for the application of the presented optimizations also for "matrix free" methods[10]), as well as the distributed memory parallelization that can be obtained in a standard way by domain decomposition[11] (in that respect, numerical integration is an embarrassingly parallel algorithm).

For our study of GPU kernel optimization, we neglect also the problem of transferring data between host (CPU) memory and GPU memory.[12] We assume that the input data are present in the main GPU memory and that the results of calculations are written to the GPU memory as well.

The limiting assumptions of our approach imply that the final decision about the feasibility of using GPUs for FEM calculations will depend upon the particular problem solved and approximation techniques used. For example, since PCIe bus, the most popular current hardware technology for connecting GPUs with host systems, is several times slower than host and GPU memory, the cost of transferring data from host memory to GPU memory has to be amortized due to particular features of problem and approximation (e.g., using the same data for different time steps or nonlinear iterations). For large scale problems, the situation gets even more complex, with the distribution of data over different computational nodes and the variety of available communication patterns for data exchange. In view of all these possible complexities, our approach is to present a strategy for achieving optimal solution for a particular phase of computations that can be employed in more comprehensive simulation scenarios.

Algorithm 1. A generic algorithm for finite-element numerical integration for a sequence of N_E elements of the same type and order of approximation

```
1:  for $e = 1$ TO $N_E$ do
2:    read problem dependent parameters
      specific to the element
3:    read geometry data for the element
4:    initialize element stiffness matrix $A^e$
5:    for $i_Q = 1$ TO $N_Q$ do
6:      compute parameters necessary to perform
        the change of variables from the reference
        element to the real element (Jacobian
        terms)
7:      calculate the corresponding discrete
        counterpart of the volume element for
        integration, vol$[i_Q]$
8:      for $i_S = 1$ TO $N_S$ do
9:        using Jacobian terms calculate the values
          of global (real) derivatives of shape
          functions and store in array $\phi[i_Q]$
10:     end for
11:     based on the problem dependent data compute
        weak form related coefficients $c[i_Q]$
12:     for $i_S = 1$ TO $N_S$ do
13:       for $j_S = 1$ TO $N_S$ do
14:         for $i_D = 0$ TO $N_D$ do
15:           for $j_D = 0$ TO $N_D$ do
16:             $A^e[i_S][j_S] += \mathbf{vol}[i_Q] \times$
                $\mathbf{c}[i_D][j_D][i_Q] \times \phi[i_D][i_S][i_Q] \times \phi[j_D][j_S][i_Q]$
17:           end for
18:         end for
19:       end for
20:     end for
21:   end for
22:   store in memory $A^e$ as the output of the
      procedure
23: end for
```

Algorithm 1 presents the version of the numerical integration algorithm that we analyze in this paper. The algorithm runs in a loop over N_E elements, with N_E assumed to be sufficiently large, to justify the use of accelerators for a given problem. The input for calculating a single element matrix is formed by a set of data defining the geometry of the element and a set of data used to compute the particular problem dependent coefficients, stored in the array c, specific to a given partial differential equation (PDE).

In its essence, the algorithm (apart from the separate calculations of Jacobian terms and real derivatives of shape functions, both related to the change of variables in the integration) is a summation for a set of output matrix entries, with two matrices used as input data. The size N_S of the local output matrix A^e is determined by the number of degrees of freedom, which, for scalar problems considered in the current paper, is equal to the number of shape functions that are used to construct finite-element solutions. Each entry in the local matrix is calculated using the values of shape functions and their derivatives, stored in arrays $\boldsymbol{\phi}$. The range of indexes i_D and j_D in $\boldsymbol{\phi}$, from 0 to N_D, is related to the fact that the arrays store the functions and their N_D partial derivatives, with N_D being the number of space dimensions of the problem (assumed to be equal three in the rest of the paper).

Each entry in the element matrix corresponds to a pair of shape functions, hence the double loop over shape functions, with indexes i_S and j_S. Each entry is obtained using quadratures, hence the outermost loop over the integration points (with index i_Q). The integration employs the change of variables (using Jacobian terms related to the mapping from the real to the reference element), hence the additional calculations of real derivatives of shape functions (at lines 8–10 of Algorithm 1) that use the geometric data for the element.

The algorithm can have different alternative variants, depending on the choice of finite-element approximations (e.g., high order,[13] discontinuous Galerkin,[14] etc.) and the problems solved. The version presented in Algorithm 1 is suitable for low-order approximations, especially for nonlinear problems, where the step of computing problem dependent coefficients has to be performed for each integration point (inducing the order of loops, where the loop over integration points is outside the loops over shape functions).

To further concentrate on the issue of performance optimization, we consider only two types of three-dimensional (3-D) finite elements (tetrahedral and prismatic), both with first order approximations. For these choices the main parameters of the algorithm that decide on the computational resource usage (the size of arrays and the number of operations), are the following:

- N_Q – the number of integration points in the selected quadrature (tetrahedra - 4, prisms - 6);
- N_S – the number of element shape functions (tetrahedra - 4, prisms - 6).

As can be seen in Algorithm 1, the actual calculations depend on the form of coefficient arrays c that can be sparse, thus, promoting some important optimizations. Usually the two innermost loops of Algorithm 1 (indexes i_D and j_D) are manually unrolled, with only the operations corresponding to nonzero terms in c performed.

We follow this approach while considering two model problems, with different forms of the coefficient array c. The first problem is the standard Poisson problem, with the Laplacian on the left hand side and some functions on the right hand side (specified by discrete values at integration points in our simulations). For this case, the coefficient array c is almost empty, with only three 1s on the main diagonal. The second case, termed as "conv–diff" problem, represents the whole group of problems with possibly complex nonlinear coefficient matrices, for which we skip the part of computations related to the particular problem solved (line 11 in Algorithm 1). By doing this, we obtain a more generic case, with the full (4×4) coefficient array c passed as input to the algorithm (we assume the array to be different for each element).

PROGRAMMING MODEL AND PARALLELIZATION OF THE ALGORITHM

In our approach, we separate two issues related to performance. First, we consider the parallelization of computations, related to the hardware environment and programming model. As the main subject of our investigations, we describe the optimization of the parallelized GPU code using autotuning.

The general hardware setting for finite-element computations using GPUs can be a single GPU, a set of GPUs, as well as a distributed memory machine, with each node equipped with one or several GPUs. In each of these cases the calculations for a single GPU are performed in one or several batches of elements, each of which is processed using Algorithm 1 (a single batch can be related to domain decomposition used in distributed memory calculations, the existence of elements with different types and orders of approximation in the finite-element mesh or the coloring technique used for avoiding data race in the assembly stage of finite-element computations[8]).

Parallelization of Finite-Element Numerical Integration

The six loops for computing the entries of A^e in Algorithm 1 present many options for parallelization. As we pointed out before, the two innermost loops are usually manually unrolled. Moreover, the loop over integration points is also usually executed by a single thread, due to the summation performed that would require some form of reduction if parallelized. The reduction overhead is absent for the loops over elements and shape functions and we briefly recall the main options for their parallelization.

One-Element-One-Thread Strategy The most obvious and always employed option for parallelization of finite-element numerical integration is to parallelize the outermost loop over elements. When this becomes the only loop that is parallelized, we get the strategy that we call *one-element-one-thread*. A single thread that is assigned to a given element performs all the calculations associated with the element. The thread can process one or several elements in a loop, in which case it executes Algorithm 1 with N_E being the number of elements assigned to it.

One-Element-Several-Threads Strategy The parallelization of any of the loops over shape functions (or both of them) can be cast into the data decomposition paradigm. The data being decomposed is A^e, and in this approach each thread operates on a set of entries of A^e. One may consider many different options in this approach, such as assigning to a given thread a small set of entries, a row or a column of A^e, or a rectangular block of entries.[8] The limiting case is to assign a single entry to a single thread.

The drawback of the naive implementation of this approach is that sequential regions appear outside the loops over shape functions. To partially remedy this, one can consider separately the parallelization of Jacobian terms calculations (the loop over shape functions for computing their real derivatives can also be parallelized, e.g., by combining it in some way with the loops for calculating A^e entries). The Jacobian terms can be computed in parallel for all integration points, which requires additional storage for intermediate results. Moreover, the optimal decomposition of calculations among threads may be different than that for calculating the entries of A^e.[15]

One-Element-Two-Kernels Strategy When fast (shared) memory resources appear too small to accommodate the intermediate data computed for all integration points, the global memory can be used for that purpose. The algorithm has to be reorganized, leading in practice to two separate kernels. In the first kernel threads calculate in parallel the necessary terms for all integration points and a set of elements (the terms may include not only Jacobian terms, but also some intermediate values related to the calculation of the problem dependent coefficients c) and stores them in the global memory. Then, the second kernel performs actual calculations of A^e entries for the elements, READING the input data from the global memory, but that time only for a single integration point at a time[16] (instead of READING the geometry and problem dependent data for an element, the precomputed data for its integration points are READ).

ALGORITHM REQUIREMENTS AND GPU HARDWARE RESOURCES

The most arithmetically intensive part of Algorithm 1 is the final evaluation of contributions to local stiffness matrices with three arrays involved:

Table 1. Number of Nonzero entries in the arrays and the computational cost of numerical integration algorithm for first order approximation, two popular types of finite elements: tetrahedral (tetra) and prismatic (prism) and two selected model problems: with Laplace operator (Poisson) and with all convection–diffusion–reaction terms (Conv–Diff).

	Type of problem			
	Poisson		Conv–diff	
	Type of element			
	Tetra	Prism	Tetra	Prism
Data for single integration point				
PDE coefficients c	0	0	16	16
Shape functions and derivatives ϕ	16	24	16	24
Total (including A^e)	**32**	**60**	**48**	**76**
Data for all integration points				
PDE coefficients c	0	0	16	16
Shape functions and derivatives ϕ	28	144	28	144
Total (including A^e)	**44**	**180**	**64**	**200**
Computational cost				
Estimated number of operations	290	2700	986	4806
Minimal number of main memory accesses	36	66	52	80
Estimated arithmetic intensity	**8**	**41**	**19**	**60**

the resulting array A^e, the coefficients $c[i_Q]$ at the current integration point (that, in principle, can be different at every point in the element), and the values of shape functions and their derivatives $\boldsymbol{\phi}[i_Q]$ at the current integration point (due to the definition of shape functions that uses the notion of the reference element, the values at integration points are the same for all considered elements, but the values of their derivatives are different for each element and, for all element types except simplexes with linear approximation, are different for each integration point).

We present the sizes of the arrays appearing in the final calculations of the entries of A^e in Table 1, for the four test cases that we consider in our computational experiments. The computational cost associated with different versions of the algorithm can only be estimated, due to possible compiler optimizations associated with such features as

- symmetry of A^e;
- constant derivatives of shape functions (the same at all integration points) for linear elements (triangles in 2-D, tetrahedra in 3-D);
- sparsity of coefficient arrays c.

For the first two features, we do not pass the relevant information directly to the compiler (nor we design the special data structures for such cases), but we inspect the assembler code created to ensure that the suitable optimizations (e.g., loop unrolling and common subexpression elimination) are performed by the compiler.

The numbers in Table 1 were verified as the minimal for produced assembler versions of the kernels used in our study. The obvious conclusion from the numbers is that the versions of the algorithm differ in their arithmetic intensity (the number of arithmetic operations per single dynamic random access memory (DRAM) memory access). Hence, e.g., the computations for the Poisson problem on tetrahedral elements are much more likely to have their performance bound by GPU memory throughput, whereas the calculations for the conv–diff problem on prismatic elements may be bound in their performance by the GPU multiprocessors computing capabilities and the shared memory throughput.

Resources of Graphics Processors

The fastest execution of Algorithm 1 can be achieved when all involved arrays are stored in registers. As an alternative, some of the data can be stored in the shared memory. The optimal choice for the placement of the variables depends on the available hardware resources and the scheduling of computations (in particular the number of threads executed simultaneously on a single multiprocessor). We present in Table 2 the

Table 2. Characteristics of GPUs used in computational experiments.

	GPU			
Architecture	Fermi	Kepler	Pascal	GCN
Graphics (accelerator) card	M2075	K20m	P100	R9 280X
Processor	GF100	GK110	GP100	Tahiti XTL
Number of multiprocessors	14	13	56	32
Global memory size [GB]	≈ 5.4	≈ 4.8	12 or 16†	3
Multiprocessor characteristics				
Number of SP/DP SIMD lanes	32/16	192/64	64/32	64/16
Number of 32 bit registers	32 768	65 536	65 536	65 536
Shared memory (SM) size [KB]	16 or 48	16 or 48	64	64
Memory resources per single SIMD lane				
Number of SP/DP registers	1024/1024	341/512	1024/1024	1024/2048
Number of SP/DP entries in SM	384/384‡	64/96‡	256/256	256/512
Performance characteristics [TFlops]				
Peak DP performance	0.515	1.17	4.7	0.87
Benchmark (DGEMM) (% of peak)	0.36 (70%)	1.10 (94%)	3.9 (83%)	0.65 (75%)
Peak SP performance	1.03	3.52	9.3	3.48
Benchmark (SGEMM) (% of peak)	0.51 (50%)	2.61 (74%)	7.8 (84%)	1.7 (49%)
Memory throughput [GB/s]				
Peak memory bandwidth	150	208	549†	288
Benchmark STREAM (% of peak)	105 (70%)	144 (70%)	380 (70%)†	219 (76%)
Machine balance				
Peak (SP/DP)	28/28	68/45	68/68†	48/24
Benchmark (SP/DP)	20/28	73/62	83/83†	31/24

† For the GP100 card there are two versions: the first with 12 GB of memory and 549 GB/s bandwidth and the second with 16 GB and 732 GB/s, we present performance characteristics only for the version with 12 GB of memory and 549 GB/s bandwidth that we used in our experiments.
‡ In maximal (48 kB) shared memory configuration.

characteristics of four graphics processors for which we test our implementations.

The first three represent three generations of NVIDIA processors targeting high performance computing (HPC) domain. The particular graphics cards, processors, and architectures used in our study are the following: M2075 card with GF100 Fermi GPU, K20m with GK110 Kepler GPU, and P100 (12 GB memory version) with GP100 Pascal GPU. The fourth GPU is a standard consumer AMD Radeon graphics card R9 280X, with Tahiti XTL processor (Graphics CoreNext, GCN, architecture).

The selection of characteristics in Table 2 concentrates on computing capabilities of processors (including the number of single instruction multiple data (SIMD) lanes for single and double precision floating point operations) and the sizes of memory pools—register files and the shared memory.

Additionally, the sizes of memory for a single SIMD lane are calculated in Table 2. This can be used to estimate how much data a single thread can store in the memory, taking into account the fine grained multithreading employed by the GPUs that requires several threads to run concurrently per SIMD lane, in order to hide latencies associated with memory accesses and pipelined execution of instructions.

The table lists also the machine balance of the considered GPUs,[17] defined as the number of floating point operations per single DRAM memory access (arithmetic intensity of the executed algorithm) required in order to reach simultaneously the peak computational performance and the DRAM memory throughput. These data can be compared with arithmetic intensities of different variants of numerical integration presented in Table 1.

OPTIMIZATION

For the purpose of investigating implementation optimization, we consider the first parallelization strategy—*one-element-one-thread*. A single thread performs Algorithm 1, operating on whole arrays associated with each assigned element. This strategy is most suitable for *h*-type finite-element method with first order approximation. However, the optimization process that we employ, together with all particular techniques and options, can be used for other types of FEM approximation, including *p*-FEM and hp-FEM. In order to adapt the process to obtain optimal kernels for different approximation types, possibly other strategies (e.g., *one-element-several-threads*) have to be applied with additional options considered (e.g., the choice of parameters for mapping the space of threads and thread workgroups onto the set of elements).

In our *one-element-one-thread* approach, we assume a single dimensional space of threads, with the minimal recommended size of workgroups for the considered GPU architectures (64 threads). This restriction results from the limited size of shared memory that is used for storing the arrays associated with the whole workgroup of threads.

Moreover, we employ minimal synchronisation necessary to ensure the correct kernel execution, design the kernels without the use of constant and texture memories and leave to the compiler final classical code optimization (including loop unrolling). We concentrate on the issues related to the placement of variables in different memory pools (shared memory, registers) and the organization of memory accesses. (In our experiments, we investigated also the padding of arrays in shared memory and found that it does not impact the performance, hence we omit this aspect.)

DRAM Access Options and the Use of Registers and Shared Memory

The flow of calculations for a single element, specified by Algorithm 1, can be divided into several phases. First, input data are READ from the main DRAM memory, then calculations are performed and, finally, the calculated output is stored back in the main memory.

We assume that the input data (geometry and PDE related) for all the elements is prepared in a single array, with the entries for a single element stored in the consecutive locations of the global DRAM memory. The recommended way of accessing DRAM memory[18] requires the threads working in lock-step (forming warps and wavefronts) to refer to memory cells in a way that allow for coalesced memory accesses.

In our case this induces the strategy, where a group of threads READS data for a sequence of elements, placing it in shared memory. Then, each thread operates on data corresponding to its assigned element, rewriting the data to registers, when required.

We also consider an alternative strategy where threads READ data on their respective elements directly from DRAM memory to registers, omitting the step of rewriting to the shared memory. We want to check whether this strategy that goes against standard practices can be better for certain architectures (despite the fact that the use of registers guarantees the best performance, READING input data to the shared memory can have additional negative impact on the performance of kernel execution, by diminishing the number of threads executed concurrently on a multiprocessor, due to limited shared memory resources).

Comparison of the algorithm requirements from Table 1 with the processor's resources in Table 2 indicates that some of, or even all, the arrays used in final calculations of A^e can be stored in the registers. We consider additionally the possibilities that one of the arrays is stored in the shared memory.

Theoretical analysis can indicate, for each considered case, the number of accesses to the arrays in different memory pools and allow for creating performance models of execution. However, the number of accesses to different levels of memory hierarchy during kernel execution will eventually depend not only on the source code, but also on compiler optimizations and the particular organization of hardware (e.g., its ability to coalesce memory accesses, the strategy of using caches or handling register spilling).

Design of Kernels for Autotuning

In order to manage the optimization of numerical integration kernels for different problems and GPUs, we design, for each finite-element type, a single, portable, parametrized OpenCL kernel[19] with several alternative options selected using a set of directives for conditional compilation.

Following the flow of computations in Algorithm 1, the first directive selects the coalesced or noncoalesced READING of the data.

For noncoalesced reading, each thread reads element input data into registers. There are two sets of the input data, the geometry data, and the problem dependent data. If any of these sets is destined to be stored in the shared memory, due to the selection done by other directives, the associated data are READ in a coalesced way.

For the options of storing arrays in the shared memory, there is a separate compiler directive for each array, i.e., the final matrix A^e, the coefficients c and the values of shape functions and their derivatives ϕ, all of which appear in final calculations of A^e entries, as well as for the array that stores geometric data (the coordinates of element's vertices in our case).

In order to design a single kernel for all the variants, the shared memory is used only for a generic, "workspace," variable in the code. Hence, the code for each particular variant is different with different variables used. The selection of the particular version to execute done by the conditional compilation directives takes care not only of using the proper arrays, but also of allocating the optimal size for arrays (this concerns especially the workspace in shared memory that in different versions of the executed code stores different variables from the algorithm). The choice for each particular option concerns several places in the source code, with many parts affected by several interfering options.

There is one additional option with the possible impact on execution performance.[16] This option is the fusion of the loop for computing the real derivatives of shape functions at an integration point (lines 8–10 of Algorithm 1) with the loops over shape functions (lines 12 and 13 in Algorithm 1). When applied, the derivatives are not computed for all shape functions and stored in some array (that uses some space in shared memory or registers) but calculated on the fly, only for the shape functions indicated by the values of indexes i_D and j_D. This option saves storage, but requires repeated calculations of real derivatives in each iteration of the inner loop over shape functions. The overhead of additional computations can be counterbalanced by performance increase if additional storage leads to register spilling or increased shared memory usage, accompanied by lower multiprocessor occupancy.

The final option, in the flow of calculations, concerns the WRITING of the computed A^e matrix entries to global DRAM memory. In the noncoalesced version, each thread WRITES the entries for its element to subsequent memory locations, so the threads executing in lock-step, forming a wavefront/warp, WRITE to memory locations with the stride equal to the size of A^e. In the coalesced version, the threads from a single wavefront/warp WRITE to subsequent memory locations, but the entries of A^e for a single element are stored with the stride equal to the number of threads in the workgroup. This requires special adaptation of other kernels performing subsequent steps of a particular FEM solution procedure (e.g., element-by-element matrix-vector product or the assembly into the global stiffness matrix) that have to know the layout of output data from numerical integration. The use of conditional compilation in the case of coalesced WRITING option is relatively straightforward, since this option interfere only with the option of possibly storing A^e in the shared memory or registers.

The final designed kernel combines all optional code variants in one source file. The flow of calculations is preserved, with the interlaced parts of the code for different execution options. The conditional compilation selects the indicated lines for each combination of directives, with a single selected block having usually from one to several lines. The number of selected lines from the source code for each particular combination of directive options does not exceed one hundred, the kernel that includes all the variants has more than one thousand lines, with more than one hundred blocks associated with different combinations of options.

As a result of our kernel design we get a set of seven options that can be specified in order to optimize the performance of Algorithm 1. For each particular selection of variants we encode the set of its options using a symbol composed of 0s and 1s, with a single position associated with particular optimization option and 0 for the optimization switched OFF and 1 for switched ON. The subsequent positions of 0s and 1s in the symbols are the following:

1. use coalesced READING of input data to the shared memory (instead of uncoalesced READING to register arrays);

2. use coalesced WRITING of output data;
3. compute all shape functions and their derivatives before entering the two innermost loops of the algorithm;
4. use shared memory for problem dependent data (PDE coefficients);
5. use shared memory for geometry data (the coordinates of element vertices);
6. use shared memory for the values of shape functions and their derivatives at integration points;
7. use shared memory for the resulting local, element arrays.

Since the last four options are mutually exclusive and we consider also the case without the use of shared memory during final calculations, with all the variables stored in registers, we get finally $2^3 * 5 = 40$ optimization variants.

COMPUTATIONAL EXPERIMENTS

We tested our OpenCL implementation of numerical integration kernels, with the described performance tuning, for the four graphics (accelerator) cards presented in Table 2. We used only double precision calculations as more versatile (we discuss the issue of precision for finite-element numerical integration e.g., in.[12,16,19]) For all considered GPUs, we used 64-bit Linux (Centos 7) with GCC version 4.8.5 (kernel 2.6.32 with CUDA 8.0, except for the Pascal GPU where we used kernel 3.10.0 and CUDA 10.1) .

We present (see Figures 1–4), for each problem and each finite-element type, the plots with the execution times for numerical integration over a single finite element (in nanoseconds), for each GPU and each combination of tuning options. In each figure, the x-axis shows the encoded tuning options, whereas the four curves correspond to different GPU architectures. The scale on the y-axis is logarithmic, due to big differences in execution times for different processors.

The organization of data on x-axis is such that the results for the five options concerning the placement of data in the shared memory (indicated by the last four digits in the encoding symbols) form the five subsequent sets of eight consecutive results, with all the combinations related to the first three options (the first three digits in the encoding symbols) within each set.

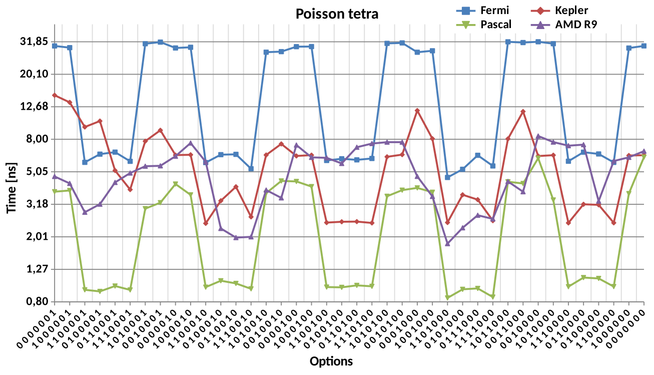


Figure 1. Execution time for numerical integration over a single finite element in the Poisson test problem, for different optimization options, tetrahedral finite elements, and the four selected GPUs.

The plots give information on hardware–software interactions for numerical integration variants and different GPUs. For the Poisson problem and tetrahedral elements, it can be seen that the most important factor for all the GPUs is the way of accessing memory. For NVIDIA processors coalesced WRITING decides on the performance. For Fermi and Pascal architectures this is the only factor, for Kepler there is additional requirement to store the output matrix in registers, not in the shared memory. For AMD R9 processor the picture is less clear, but the best results are obtained for storing PDE coefficients or shape function arrays in the shared memory and using coalesced READING and WRITING.

For the Poisson problem and prismatic elements the calculations play more important role than for tetrahedral elements. Hence, the use of shared memory, instead of registers, for storing the output element matrix usually leads to slower

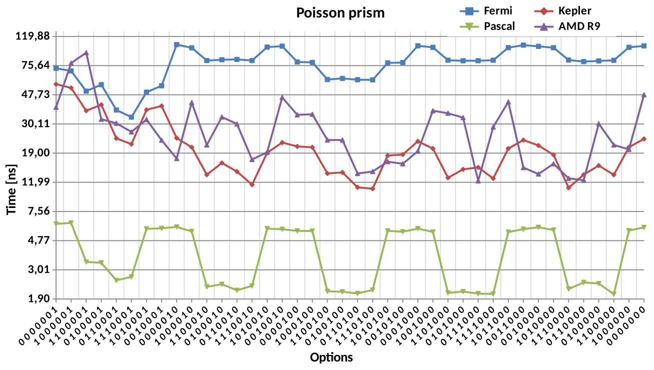


Figure 2. Execution time for numerical integration over a single finite element in the Poisson test problem, for different optimization options, prismatic finite elements, and the four selected GPUs.

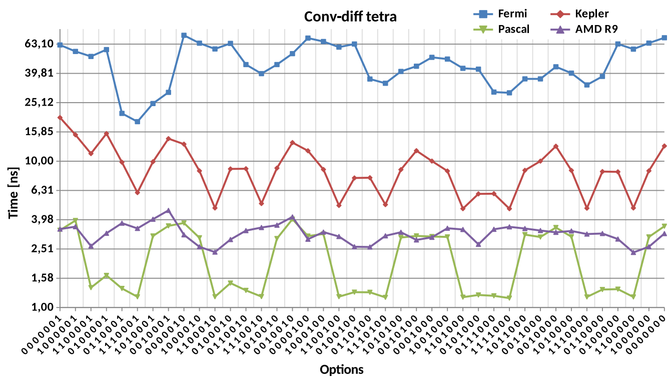


Figure 3. Execution time for numerical integration over a single finite element in the convection–diffusion test problem, for different optimization options, tetrahedral finite elements, and the four selected GPUs.

execution. The only exception is the Fermi architecture, for which storing the large output matrix in the shared memory is the only way to significantly reduce register spilling that slows down the execution. Once again, to achieve the best performance, coalesced READING and WRITING should be employed for all the architectures, but now there are other factors that has to be taken into account. These factors are different for each architecture and are reflected by different shapes of execution profiles. The most random is the profile for AMD R9 processor, whereas Pascal GP100 shows that its massive computational resources are sufficient to make coalesced WRITING the only factor significantly influencing the performance.

The same profile for the Pascal architecture appears also for the conv–diff problem on tetrahedral elements. For this problem, however, the other architectures show more interesting profiles. Kepler GPU requires both, coalesced READING and WRITING, but without using shared memory for the output stiffness matrix. On the contrary, the Fermi processor requires storing the output matrix in the shared memory (and computing all global derivatives of shape functions at once), due to its problems with providing sufficient number of registers. The best options for Fermi belong to the worst options for AMD R9. For this GPU, one of the best results is achieved for not using the shared memory at all—one can see here the influence of the large number of double precision registers for a single SIMD lane for this architecture.

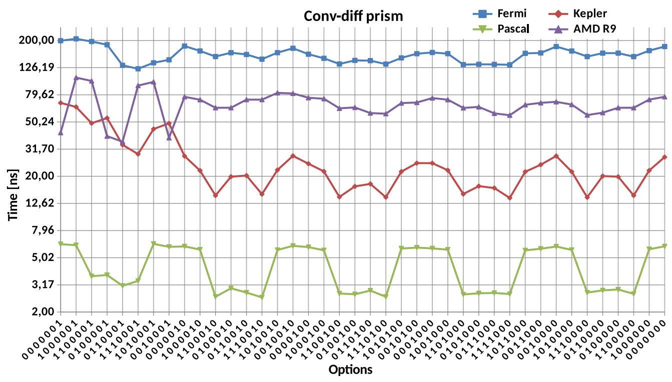


Figure 4. Execution time for numerical integration over a single finite element in the convection–diffusion test problem, for different optimization options, prismatic finite elements, and the four selected GPUs.

The profile for R9 GPU and conv–diff problem on prismatic elements shows how dramatically the performance of the processor can decrease for kernels with larger memory requirements. The R9 Tahiti GPU is faster than the Kepler GPU for the conv–diff problem and tetrahedral elements but is much slower for prismatic elements. The best results for R9 are obtained for storing the output matrix in the shared memory that clearly indicates problems with providing enough registers for faster calculations. Surprisingly, one of the best results is obtained for noncoalesced READING and WRITING, which means that computations take for this case more time than explicit memory accesses. The problems with memory resources for R9 are indicated also by the fact that coalesced READING that requires additional shared memory, significantly slows down the calculations with the output matrix in shared memory.

The Fermi processor has relatively flat profile for the conv–diff problem on prismatic elements, with no one option slowing down significantly the execution. This is not true for Kepler, where storing the output matrix in the shared memory slows down the execution. The Pascal processor has almost the same profile as for the other test cases, its high computational power with respect to memory throughput manifests itself with hiding computation time by memory access time.

Scalability

In order to obtain performance characteristics of the developed kernels related to scalability, we performed experiments for different numbers of threads and different numbers of elements. We present the results for the newest and specifically designed for HPC architecture, Pascal GP100, and a selected kernel that performed well for all the test cases—the one with coalesced READING and WRITING, as well as computing all shape functions in advance and storing PDE related data in the shared memory. Instead of relatively small meshes used for the test examples designed for all considered GPUs (with 705 894 tetrahedral and 97 792 prismatic elements), we considered the meshes with up to 5 647 152 tetrahedral and 6 258 688 prismatic elements. We obtained good scaling with respect to the number of elements that confirmed the results reported in our previous publications.[20]

We present the performance data for the largest meshes that fitted only the large memory of Pascal GPU. First, we show the classical strong scalability results in Figure 5, with the number of thread workgroups as the measure of the number of threads. It can be seen that, since we limit, our study only to a single GPU, the speed-up curves depart from the perfect speed-up for the growing number of threads. Nevertheless, for the number of workgroups up to 56 (one workgoup per GPU multiprocessors), we get good speed-up (with the parallel efficiency above 70%). Moreover, with the increasing arithmetic intensity of the kernels, the speed-up curves get closer to the perfect speed-up.

Since the speed-up decreases for the larger numbers of threads, we present the data that indicate how the absolute performance of the kernel behaves in that case. Figure 6 shows the absolute performance in GFlops for all test cases as the function of the number of thread workgroups, up to four workgroups per GPU multiprocessor.

Further increase of the number of threads did not improve the results that remain at the level 8%–57% of the theoretical maximum. Since the computational capabilities of the GPU are not fully exploited, we checked the utilization of memory bandwidth. The summary of the results is presented in Figure 7 that shows the performance of the kernel put on the roofline graph[21] for the Pascal GPU. The kernel achieves memory throughput, close to the throughput obtained with the STREAM benchmark, indicating that it approaches the available optimum.

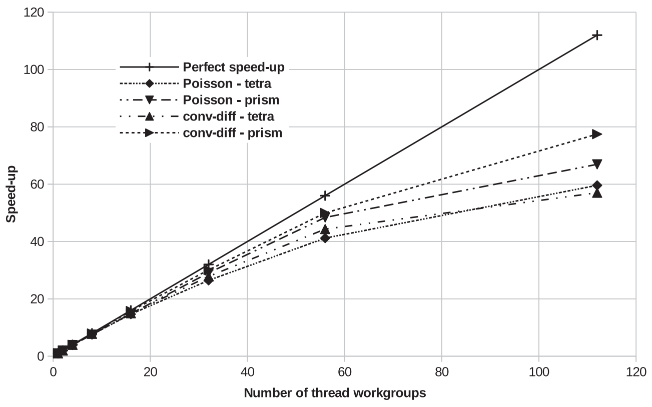


Figure 5. Performance characteristics for Pascal GP100 processor—speed up for the increasing number of thread workgroups (up to two per GPU multiprocessor) and the four test cases.

Optimal Kernels

As the final result of our autotuning the best versions for each kernel and each GPU architecture were selected for the meshes with 705 894 tetrahedral and 97 792 prismatic elements and their performance results presented in Table 3, together with the results on the large mesh with 5 647 152 tetrahedral and 6 258 688 prismatic elements, obtained with the kernel selected for the scalability study.

We used the execution time results to calculate the estimated performance characteristics in terms of the number of floating point operations per second (in TFlops) and the DRAM memory throughput (in GB/s). Additionally, in the same Table 3, we compared the

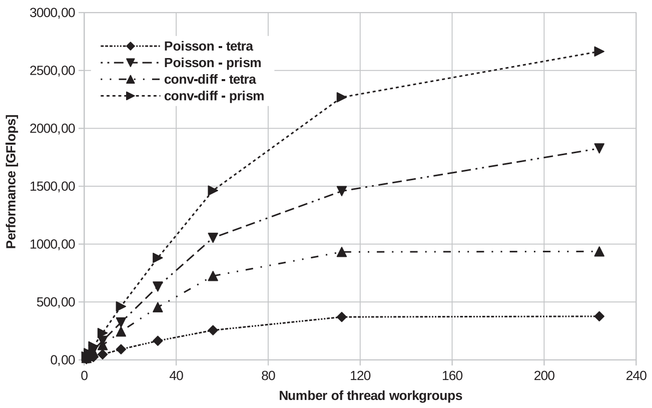


Figure 6. Performance characteristics for Pascal GP100 processor—performance in GFlops for the increasing number of thread workgroups (up to four per GPU multiprocessor) and the four test cases.

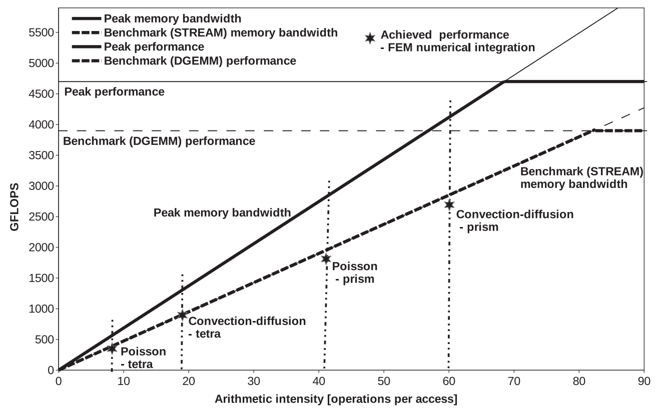


Figure 7. Performance characteristics for Pascal GP100 processor—roofline model diagram with the performance achieved by a selected numerical integration kernel for the four test cases.

performance achieved in our experiments with the standard performance characteristics of the GPUs—their theoretical peak and typical benchmark data in TFlops and GB/s, presented in Table 2.

It can be seen that the computations for tetrahedral elements, with low arithmetic intensity, stress much more the memory systems of GPUs. The performance of considered processors is much below their computing capabilities, but shows significant usage of their memory bandwidth. All the GPUs handle well the case of numerical integration for the Poisson problem on tetrahedral elements with the throughput of DRAM memory above 40% of the theoretical peak. For the conv–diff problem, only Fermi architecture decreases its performance, due to the problems with too small number of registers available to kernels that causes low occupancy of multiprocessors.

For the conv–diff problem, with higher arithmetic intensity, reaching satisfactory performance turned out to be difficult for all the GPUs except Pascal GP100. Its best results, presented in Table 3 and Figure 7, should be considered satisfactory for a relatively complex algorithm of finite-element numerical integration.

CONCLUSION

This article presented the use of automatic experimental tuning for approaching the optimal execution performance for the algorithm of finite-element numerical integration and selected GPU architectures. For each tested GPU and all the variants of finite-element numerical integration, we obtained different (sometimes slightly, sometimes largely) execution profiles with different, sometimes not intuitively obvious, optimization options optimal for different GPU architectures.

In our case of optimizing finite-element numerical integration, the most important aspect was the selection of variables to put in different memory pools, associated with the necessary organization of computations. We have shown that the proper selection of tuning options and its implementation

Table 3. Performance results for the best variants of numerical integration for each GPU architecture, two types of finite elements with first order approximation: tetrahedral (tetra) and prismatic (prism) and the two model problems: with Laplace operator (Poisson) and with all convection–diffusion–reaction terms (conv–diff).

Type of element	Tetra					Prism				
Mesh size	Number of elements 705 894				5 647 152	Number of elements 97 792				6 258 688
GPU	Fermi	Kepler	Tahiti	Pascal		Fermi	Kepler	Tahiti	Pascal	
				Poisson problem						
Execution time [ns]	4.66	2.24	1.82	0.84	0.77	33.52	11.63	12.26	2.04	1.48
Estimated GFlops	62	129	159	336	375	80	232	219	1317	1826
% peak GFlops	12	11	18	7	8	16	20	25	28	38
Minimal GB/s	62	128	158	333	373	15	45	43	257	357
% peak GB/s	41	62	55	61	68	11	22	15	47	65
				Conv–diff problem						
Execution time [ns]	18.61	5.13	2.38	1.15	1.05	123.83	14.36	35.97	2.56	1.80
Estimated GFlops	53	192	414	826	936	38	334	134	1874	2664
% peak GFlops	10	16	47	18	20	8	29	15	40	57
Minimal GB/s	22	81	175	348	395	5	44	18	249	355
% peak GB/s	15	39	61	64	72	3	21	6	45	65

require deep analysis of the algorithm in view of the available hardware resources.

As extensions of the presented investigations and the directions for further research, we plan to consider in forthcoming publications vector, nonlinear problems, second order and mixed approximations, additional options, and parameters for autotuning as well as different parallelization strategies (especially *one-element-several-threads*).

REFERENCES

1. Y. Zhang and J. Owens, "A quantitative performance analysis model for GPU architectures," in *Proc. IEEE 17th Int. Symp. High Perform. Comput. Archit.*, Feb. 2011, pp. 382–393.
2. J. Sim, A. Dasgupta, H. Kim, and R. Vuduc, "GPUPerf: A performance analysis framework for identifying potential benefits in GPGPU applications," in *Proc. 17th ACM SIGPLAN Symp. Princ. Pract. Parallel Program.*, 2012, pp. 11–22, doi: 10.1145/2145816.2145819.
3. K. Banaś, F. Krużel, J. Bielański, and K. Chłoń, "A comparison of performance tuning process for different generations of NVIDIA GPUs and an example scientific computing algorithm," in *Parallel Processing and Applied Mathematics—12th International Conference, PPAM 2017*, Lublin, Poland, September 10–13, 2017, Revised Selected Papers, Part I (Lecture Notes in Computer Science, 10777), R. Wyrzykowski, J. J. Dongarra, E. Deelman, and K. Karczewski, Eds., Berlin, Germany: Springer, 2017, pp. 232–242, doi: 10.1007/978-3-319-78024-5_21.
4. A. Sussman, "Model-driven mapping onto distributed memory parallel computers," in *Proc. Proc. ACM/IEEE Conf. Supercomput.*, 1992, pp. 818–829, doi: 10.1109/SUPERC.1992.236681.
5. A. Davidson and J. Owens, "Toward techniques for auto-tuning GPU algorithms," in *Proc. 10th Int. Conf. Applied Parallel Sci. Comput.*, 2012, vol. 2, pp. 110–119, doi: 10.1007/978-3-642-28145-7_11.
6. M. Tillmann, T. Karcher, C. Dachsbacher, and W. F. Tichy, "Application-independent autotuning for GPUs," in *Proc. Parallel Comput., Accelerating Comput. Sci. Eng., Int. Conf. Parallel Comput.*, Sep. 2013, pp. 626–635.
7. P. Płaszewski, P. Macioł, and K. Banaś, "Finite element numerical integration on GPUs," in *Proc. 8th Int. Conf. Parallel Process. Appl. Math.*, 2010, pp. 411–420.
8. K. Banaś, F. Krużel, J. Bielański, and K. Chłoń, " Finite element integration and assembly on modern multi and many-core processors," in *Advances in Parallel, Distributed, Grid and Cloud Computing for Engineering*, P. Iványi, B. Topping, and G. Várady, Eds. Stirlingshire, U.K.: Saxe-Coburg Publications, 2017, pp. 51–78.
9. C. Cecka, A. J. Lew, and E. Darve, "Assembly of finite element methods on graphics processors," *Int. J. Numer. Methods Eng.*, vol. 85, no. 5, pp. 640–669, 2011, doi: 10.1002/nme.2989.
10. K. Ljungkvist, "Matrix-free finite-element computations on graphics processors with adaptively refined unstructured meshes," in *Proc. 25th High Perform. Comput. Symp.*, Apr. 2017, pp. 1:1–1:12. [Online]. Available: http://dl.acm.org/citation.cfm?id=3108097
11. D. Komatitsch, G. Erlebacher, D. Göddeke, and D. Michéa, "High-order finite-element seismic wave propagation modeling with MPI on a large GPU cluster," *J. Comput. Phys.*, vol. 229, no. 20, pp. 7692–7714, 2010.
12. K. Banaś, F. Krużel, and J. Bielański, "Finite element numerical integration for first order approximations on multi- and many-core architectures," *Comput. Methods Appl. Mech. Eng.*, vol. 305, pp. 827–848, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045782516301256
13. A. Dziekonski, P. Sypek, A. Lamecki, and M. Mrozowski, "Finite element matrix generation on a GPU," *Progress Electromagn. Res.*, vol. 128, pp. 249–265, 2012.
14. A. Klöckner, T. Warburton, J. Bridge, and J. S. Hesthaven, "Nodal discontinuous Galerkin methods on graphics processors," *J. Comput. Phys.*, vol. 228, pp. 7863–7882, Nov. 2009.
15. M. G. Knepley, K. Rupp, and A. R. Terrel, "Finite element integration with quadrature on the GPU," *CoRR*, vol. abs/1607.04245, 2016. [Online]. Available: http://arxiv.org/abs/1607.04245
16. K. Banaś, P. Płaszewski, and P. Macioł, "Numerical integration on GPUs for higher order finite elements," *Comput. Math. Appl.*, vol. 67, no. 6, pp. 1319–1344, 2014.
17. J. D. McCalpin, "Memory bandwidth and machine balance in current high performance computers," *IEEE Comput. Soc. Tech. Committee Comput. Archit. Newslett.*, vol. 12, pp. 19–25, Dec. 1995.
18. *NVIDIA CUDA C Programming Guide Version 5.0*, NVIDIA, 2012.

19. K. Banaś and F. Krużel, “OpenCL performance portability for xeon phi coprocessor and NVIDIA GPUs: A case study of finite element numerical integration,” in *Proc. Eur. Conf. Parallel Process.*, Aug. 25–26, 2014, Revised Selected Papers, Part II, (Lecture Notes in Computer Science, 8806), Berlin, Germany: Springer, 2014, pp. 158–169, doi: 10.1007/978-3-319-14313-2_14.
20. P. Płaszewski, K. Banaś, and P. Macioł, “Higher order FEM numerical integration on GPUs with OpenCL,” in *Proc. Int. Multiconf. Comput. Sci. Inf. Technol.*, Oct. 2010, pp. 337–342.
21. S. Williams, A. Waterman, and D. Patterson, “Roofline: An insightful visual performance model for multicore architectures,” *Commun. ACM*, vol. 52, no. 4, pp. 65–76, Apr. 2009, doi: 10.1145/1498765.1498785.

Krzysztof Banaś is a professor with the AGH University of Science and Technology, Cracow, Poland. His research interests include high performance computing, computational aspects of the finite element method, and GPGPU computing. He received the M.Sc. degree in applied mechanics from the Cracow University of Technology and in mathematics from Warwick University, Coventry, U.K., and the Ph.D. degree from the Institute of the Fundamental Technological Research of the Polish Academy of Sciences, concerning finite elements in CFD. He is the corresponding author of this article. Contact him at pobanas@cyf-kr.edu.pl.

Filip Krużel is an assistant professor with the Institute of Computer Science of the Cracow University of Technology. His research interests include high performance computing and the use of various types of accelerators and nonstandard hardware architectures. He received the Ph.D. in computer science from the Institute of the Fundamental Technological Research of the Polish Academy of Sciences in 2018. Contact him at fkruzel@pk.edu.pl (http://www.pk.edu.pl/ fkruzel).

Jan Bielański is a research and teaching assistant with the AGH University of Science and Technology and is working toward the Ph.D. degree with Jagiellonian University. His research interests include finite element method, parallel computing, GPGPU computing, heterogeneous computing, and high performance computing. He received the B.Sc. and M.Sc. degrees in computer science, with distinction, from the Cracow University of Technology, Poland, in 2010 and 2012, respectively. Contact him at jbielan@agh.edu.pl.

Editors: James J. Hack, jhack@ornl.gov
Michael E. Papka, papka@anl.gov

The U.S. High-Performance Computing Consortium in the Fight Against COVID-19

James J. Hack
Oak Ridge National Laboratory

Michael E. Papka
Argonne National Laboratory

***Abstract*—U.S. computing leaders, including Department of Energy National Laboratories, have partnered with universities, government agencies, and the private sector to research responses to COVID-19, providing an unprecedented collection of resources that include some of the fastest computers in the world. For HPC users, these leadership machines will drive the AI to accelerate the discovery of promising treatments, enable at-scale simulations to understand the virus's protein structure and attack mechanisms, and help inform policymakers to deploy resources effectively.**

Beginning at the time the World Health Organization declared COVID-19 a pandemic in mid-March 2020, the research community has undertaken an immense global collaboration to suppress SARS-CoV-2, the novel coronavirus that causes the disease. The two biggest objectives in COVID-19 research: development and evaluation of vaccines and therapeutic treatments, and predicting possible trajectories of future outbreaks by developing a clear understanding of how this disease is transmitted.

Governments and public health officials worldwide are struggling to control a virus that is already widespread and easily transmissible, including by asymptomatic carriers. Countries with the advanced research facilities and infrastructure, ready experimental pipelines, and the participation of partnerships between entities such as biotechnology companies, academic

Digital Object Identifier 10.1109/MCSE.2020.3019744
Date of current version 9 October 2020.

research laboratories, and government national laboratories—all of which the U.S. has rapidly put in place—have a distinct advantage in shaping that response.

Early in the battle to understand, treat, and control the COVID-19 pathogen, a unique private-public effort was established, led by the White House Office of Science and Technology Policy, the U.S. Department of Energy (DOE), and IBM to bring together federal government, industry, and academic leaders who would make computational resources available to the COVID research community. The goal of the *COVID-19 High Performance Computing Consortium* ("The HPC Consortium") was to help accelerate the pace of scientific discoveries in COVID-19 research projects.

The consortium brings together more than 30 partners from 7 national labs; NASA; 8 National Science Foundation-funded organizations, notably XSEDE, which is processing the requests for consortium resources, and the University of Texas at Austin's Texas Advanced Computing Center, one of the consortium's leading providers of computing cycles; 11 technology companies and 14 academic institutions, and provides resources ranging from small clusters to cloud and web services to high-end supercomputers.

The DOE's Office of Science operates many of the world's largest user facilities enabling the most cutting-edge coronavirus research studies today, such as efforts to understand its structure and underlying biochemistry, and evaluating possible therapeutics. Among those user facilities are DOE's Leadership Computing Facilities, located at Oak Ridge National Laboratory and Argonne National Laboratory. Oak Ridge's Summit IBM supercomputer and Argonne's Theta Intel Cray supercomputer are currently enabling the most cutting-edge coronavirus research studies today, powering research in molecular modeling, bioinformatics, and epidemiology to help accelerate the development of treatments and strategies to combat the COVID-19 pandemic. Access to these unique computational ecosystems has been made available through proposal mechanisms established by the HPC Consortium where the capabilities include access to state-of-the-art modeling algorithms, data analysis tools, and huge numbers of CPUs and GPUs. While relatively few consortium members are able to exploit all of these capabilities, the inclusion of leadership computing machines in this arsenal represents a significant commitment of technologies that are suited to run simulations at the scale necessary to significantly accelerate knowledge on how to control a global epidemic, and to address rapidly evolving advances in our understanding of the virus. Consortium projects include running complex biophysics investigations at scale to help understand what treatment protocols are and are not working; rapidly screening for possible therapeutics using machine learning and deep learning techniques; and running agent-based models to run scenarios for how the virus will spread given evolving guidance on social distancing and other hygienic behavior.

This article highlights several HPC Consortium projects, and other related activities, actively being supported at DOE's Leadership Computing Facilities and how these machines are being used to accelerate the science needed to develop treatments and strategies to combat COVID-19.

COVID-19 PUTS HIGH-END COMPUTING ARCHITECTURES TO THE TEST

Summit's large array of very powerful hybrid CPU-GPU nodes—more than 4600 nodes, each containing 44 Power 9 cores and 6 NVIDIA Volta GPUs—make the platform especially well-suited to some specific but important types of investigations related to the pandemic. First, molecular dynamics (MD) simulations of the virus and its interactions with human cells can be carried out on Summit at scales and speeds that are unobtainable on other resources. In addition, the use of machine learning techniques to study how potential therapeutics interact with one another, and with other drug compounds, can take advantage of the unique AI features of Summit's GPU architecture.

Theta is a Cray XC40 manycore system built on the Intel KNL 7230 processor with 64 cores each. The system is comprised of 4392 nodes for a total core count of 281 088. The system is well suited for efforts that rely on the x86 instruction set. In addition, Argonne has enabled near-real-time response queues for Theta by exploiting drain time of the system—this allows for coupling with other unique instruments like Argonne's Advanced Photon Source (APS) for on-demand analysis. This capability, along with Theta's traditional support for modeling and simulation, make it an important resource in supporting research for the pandemic.

COMPUTATIONAL ADVANCES IN EPIDEMIOLOGICAL MODELING

- Argonne computational scientist *Jonathan Ozik* and his team have developed CityCOVID, an agent-based model capable of tracking detailed COVID-19 transmission. Agent-based modeling is an approach for capturing the dynamics of heterogeneous, interacting, adaptive agents at an individual, granular level of detail. When applied to a city such as Chicago, CityCOVID includes a synthetic population representing the 2.7 million residents of Chicago and the 1.2 million geolocated places where they can colocate. Throughout a simulated day, each individual, or agent, moves from place-to-place, hour-by-hour, engaging in social activities and interactions with colocated agents, where COVID-19 exposure events can occur. The COVID-19 disease progression is modeled within each individual, including differing symptom severities, hospitalizations, and age-dependent probabilities of transitions between disease stages. Using Argonne computing resources, CityCOVID is being used to calibrate unobserved model parameters, such as the time-varying degree of individual self-protective behaviors across the population, and to simulate a variety of interventions and future scenarios. While the Argonne team has been using Chicago as a testbed for developing these capabilities, CityCOVID is being extended to other regions as well.
- University of West Florida computer science professor *Ashok Srinivasan* is using Argonne resources to analyze the spread of COVID-19 during air travel through pedestrian dynamics simulation. Srinivasan's work attempts to better understand the impact of new boarding processes on exposure to the virus. To do that, Srinivasan developed a computationally efficient constrained linear movement model and code and carried out a large parameter space sweep to simulate realistic boarding scenarios. The simulation results show that back-to-front boarding roughly doubles the infection exposure compared with random boarding. It also increases exposure by around 50% compared to a typical boarding process prior to the outbreak of COVID-19.
- Purdue University professor *Yung-Hsiang Lu* and a team of engineers built a website that pools together live footage and images from approximately 30 000 network cameras in more than 100 countries to study social distancing, making data easier to analyze. The site has documented footage since March 2020 that could help evaluate the effectiveness of lockdowns and restrictions. The project uses Argonne computational resources and storage.

ADVANCED MODELING AND SIMULATION

- *G. Andrés Cisneros*, at the University of North Texas, is using Summit to investigate inhibitor mechanisms of existing drugs for the RNA dependent RNA polymerase (RDRP) protein to provide insights that could serve to improve treatment options for COVID-19. He and his team are using classical MD techniques to investigate the structure and dynamics of the protein with and without inhibitors. He is also looking at Remdesivir, a broad-spectrum antiviral medication, and several other drugs, to understand how these protein inhibitors are useful in battling COVID-19 infections. Since protein inhibitors can be used to prevent viruses and bacteria from reproducing, their plan is to study the interactions and reaction mechanisms at the atomic level, something that cannot be done in a traditional lab.

RAPID ANALYSIS COUPLED WITH EXPERIMENTAL REFINEMENT

- A team of researchers at Oak Ridge and the University of Tennessee, Knoxville (UTK) led by *Jeremy Smith*, has already used Summit to build a model of the coronavirus' spike protein, also called the S-protein, based on early studies of the protein structure. Using MD simulations, they explored approximately 800 000 different compounds' docking properties to the spike to determine if any might prevent it from binding to human cells. They ultimately identified 77 small-molecule drug compounds—medications, natural compounds, etc.—shown by the simulations to bind to regions of the spike that are important for entry into the human cell and, therefore, might interfere with the infection process. The team then took this work further using more complete descriptions of the virus structure, one of which was determined using x-ray crystallography by a team from the Spallation Neutron Source at Oak Ridge and a team at Argonne's APS. In a first-of-its-kind measurement, the protease was determined from room-temperature crystallized protein samples to achieve a more accurate three-dimensional model of the protein (see Figure 1). This provided the opportunity to computationally explore compounds that can be shown to bind to the protein, which might block the virus's replication mechanism. Using the improved protein structures, the Smith team performed MD simulations to computationally screen 1.5 billion chemical compounds for COVID-19 in only 24 h, providing a wealth of data for SARS-CoV-2 drug discovery efforts. Because the virus's main protease is an important drug target, the simulations might lead to an understanding of which drugs could be successfully repurposed for COVID-19.

- Scientists from Argonne's Structural Biology Center at the APS, including Argonne Distinguished Fellow *Andrzej Joachimiak*, and working with *Mateusz Wilamowski* of the Center for Structural Genomics of Infectious Diseases (CSGID), performed experiments on the Nsp10+Nsp16 protein complex of SARS-CoV-2 created at CSGID. The experiment provided the first low-dose, room-temperature insight into the structure of this protein complex. Its results will give the community greater biological insight into the complex than is possible with traditional crystallography techniques. The experiment is designed to determine the metal activation of the complex and later lead to time-resolved experiments, which will be the first dynamic structural experiment of a SARS-CoV-2 related protein. Time-resolved structural dynamics will help elucidate the electrochemistry of this protein function and give insights into the virus. To support the rapid processing requirements, a team led by computer scientists *Ryan Chard* and beamline scientist *Darren Sherrell* deployed an automated data acquisition, analysis, curation, and visualization pipeline, leveraging Theta for high-speed on-demand analysis. The pipeline reactively analyzes data as it is collected, moving images of the sample from the APS to the Argonne Leadership Computing Facility where they are rapidly analyzed and visualized. The same automated pipeline then moved results to a repository and extracted metadata for publication in a data portal, which scientists can monitor during an experiment.
- A team led by *Albert Lau*, assistant professor of biophysics and biophysical chemistry at the Johns Hopkins School of Medicine, is using a joint computational and experimental approach to identify FDA-approved drugs that possess antiviral activity against SARS-CoV-2. Theta is being used to screen a library of compounds that includes existing FDA-approved drugs against ten intracellular catalytic SARS-CoV-2 protein targets for

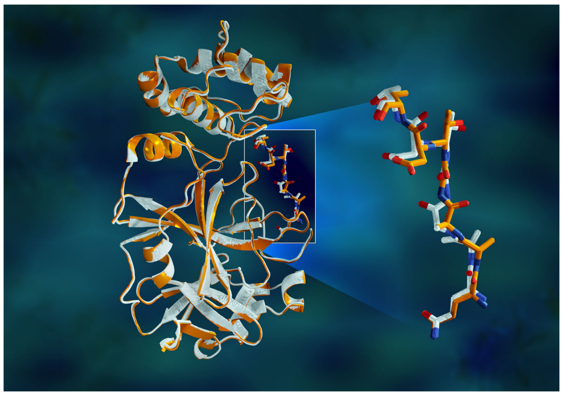


Figure 1. Overlapping x-ray data of the SARS-CoV-2 main protease shows structural differences between the protein at room temperature (orange) and the cryogenically frozen structure (white). This work is being used by Jeremy Smith and his team who are conducting drug docking simulations using Summit. Credit: Jill Hemman/ORNL, U.S. Dept. of Energy.

binding, with the goal of identifying drugs that disrupt viral protein function and diminish viral viability. Reducing the number of compounds and protein targets to an experimentally manageable number will help the team establish priorities for the subsequent workflow, in which target proteins will be expressed and purified, and activity assays will be developed. Finally, prioritized compounds in the Johns Hopkins Drug Library will be screened against the purified target proteins to identify those that exhibit antiviral activity.

- A multi-institutional team from the UTK, the Yale School of Medicine, the U.S. Department of Veterans Affairs, the Versiti Blood Research Institute, the University of Kentucky and the Cincinnati Children's Hospital Medical Center, led by Oak Ridge computational biologist *Dan Jacobson*, has performed data analyses on Summit to analyze samples of lung fluid cells from COVID-19 patients. Analysis of COVID and control patients is aimed at finding gene expression and coexpression patterns that may explain the runaway symptoms produced by the body's response to SARS-CoV-2. The team required the power of Summit to run 2.5 billion correlation calculations that helped them understand the normal regulatory circuits and relationships for the genes of interest. With Summit, the team completed the calculations in one week rather than spending months doing them on a desktop computer. Early results have found that genes related to one of the body's systems responsible for lowering blood pressure—the bradykinin system—appear to be excessively "turned on" in the lung fluid cells of those with the virus. A bradykinin storm could explain the wide variety of symptoms experienced by COVID-19 patients, such as muscle pain, fatigue, nausea, vomiting, diarrhea, headaches, and decreased cognitive function. At least ten existing drugs are known to act on the specific pathways the team studied, but large-scale clinical trials are needed to determine whether they might be effective at treating COVID-19.

A bradykinin storm could explain the wide variety of symptoms experienced by COVID-19 patients.

RAPID SCREENING USING AI TECHNIQUES

- The many clinical manifestations of COVID-19 have so far proven to be resistant to existing single drug therapies. A team led by *Jennifer Diaz* of the Icahn School of Medicine at Mount Sinai has developed a machine learning classifier to predict whether certain drug combinations might be effective against the virus. The team is using the classifier on Summit to predict gene expression patterns for more than 700 000 combinations of existing drugs. In a preliminary analysis, Diaz's team identified U.S. Food and Drug Administration (FDA)-approved drugs that are already being studied to treat COVID-19 as well as novel drugs that may have a synergistic effect when combined. Studies have suggested these synergistic gene expression predictions can be used to identify biological pathways and processes that will be altered for each combination. The team plans to expand the database, improve the classifier, and use gene set enrichment analysis to make predictions of drug pairs that may be a synergetic COVID-19 treatment. Predictions from this unique analysis must be validated in wet labs or clinical trials to determine their viability and efficacy. Successful laboratory validation will allow a broadening of the project scope to explore highly focused drug combinations in the millions.
- An Argonne-based team is seeking to address both the fundamental biological mechanisms of the virus and the disease, while simultaneously targeting the entire viral proteome to identify potential therapeutics. Argonne computational biologist *Arvind Ramanathan* and Argonne Associate Laboratory Director *Rick Stevens*, are leveraging Argonne and Oak Ridge leadership machines to design novel

therapeutics against SARS-CoV-2 using AI approaches that integrate information from experimental observations, and rigorous, physics-based virtual screening and molecular simulations, to identify viable drugs that can inhibit viral proteins. These AI approaches, based on advances in deep learning and reinforcement learning, are capable of predicting how strongly a small molecule will bind to a protein as well as explore the structural space of compounds that are predicted to bind to find more suitable variants.

The database of potential drug candidates for COVID-19 is immense, including millions to billions of potential compounds. So far, computational screening of small molecules has resulted in identifying molecules that can potentially inhibit viral function in wet lab experiments. These experiments, performed at the Argonne-located BSL-3 facility, involve live human lung cell cultures being exposed to small molecules followed by subsequent measurements that monitor viral replication. These molecules are being further refined to optimize them for binding to specific viral target proteins. Using AI techniques, the team has screened over 6 million small molecules and are validating them at Argonne for activity against the virus and rapidly expanding it to screen billions of compounds. The potential impact of this work is the design of new generative models based on reinforcement learning for both small molecules and antibodies; and development of large-scale, AI-driven simulations of the entire viral particle and drugs bound to the various viral targets, as a better pathway to an antiviral drug.

ACKNOWLEDGMENTS

This article has been authored by UT-Battelle, LLC, operator of Oak Ridge National Laboratory, a U.S. Department of Energy Office of Science Laboratory, operated under contract number DE-AC05-00OR22725 and by UChicago Argonne, LLC, operator of Argonne National Laboratory, a U.S. Department of Energy Office of Science Laboratory, operated under contract number DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan. http://energy.gov/downloads/doe-public-access-plan. The authors wish to acknowledge the contributions of Laura Wolf, Argonne National Laboratory, in the preparation of this article.

James J. Hack is currently the ORNL Computing and Computational Science Director of HPC Strategies, and was the former Director of the National Center for Computational Sciences at Oak Ridge National Laboratory, which houses the Oak Ridge Leadership Computing Facility, from 2007 to 2019. After earning a Ph.D. degree in atmospheric dynamics from Colorado State University, he was a research staff member at the IBM T. J. Watson Research Center working on HPC architectures, later moving to the National Center for Atmospheric Research, where he was a leader in the development of NCAR's global atmospheric model. His primary scientific interests include physical parameterization techniques, numerical methods and their implementation on HPC platforms, and diagnostic methods for evaluating simulation quality. A former editor of the *Journal of Climate*, he has testified before Congress on the topic of climate change, and has been involved in National Research Council studies on climate modeling. Contact him at jhack@ornl.gov.

Michael E. Papka is a senior scientist and member of the senior leadership at Argonne National Laboratory. He joined Argonne in 1992 and has served as Deputy Associate Laboratory Director for Computing, Environment and Life Sciences since 2006, and also as Director of the Argonne Leadership Computing Facility since 2010. He received the master's and the Ph.D. degrees in computer science from the University of Chicago. His research interests span visualization, large-scale data analysis, and the development and deployment of research infrastructure in support of science. He is a Presidential Research, Scholarship and Artistry Professor in the Department of Computer Science at Northern Illinois University and leads the Data, Devices, and Interaction Laboratory (ddiLab). Contact him at papka@anl.gov.

Data Visualization for the Understanding of COVID-19

João L. D. Comba
Universidade Federal do Rio Grande do Sul

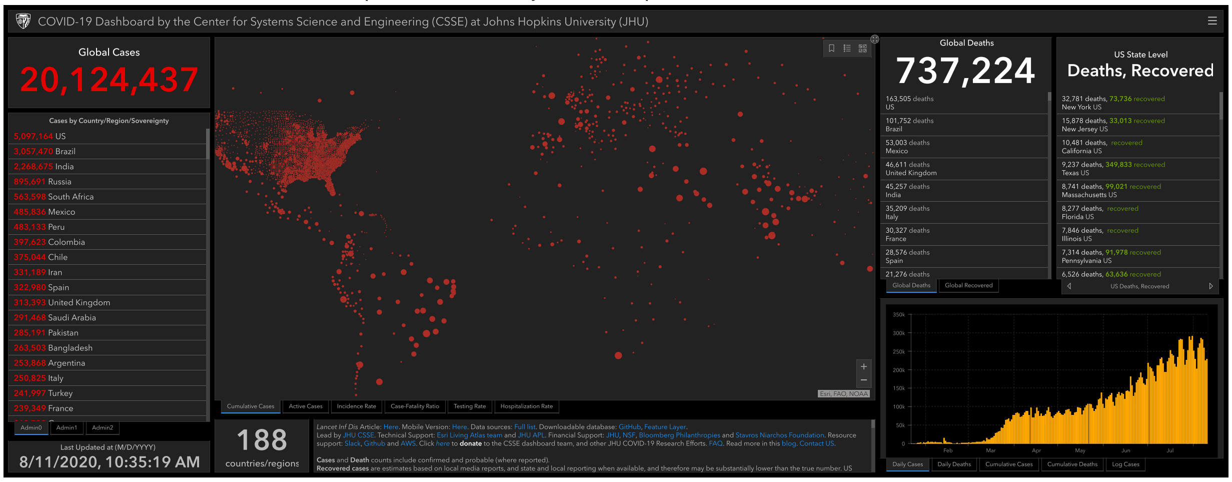
***Abstract*—Visualization techniques have been front-and-center in the efforts to communicate the science around COVID-19 to the very broad audience of policy makers, scientists, healthcare providers, and the general public. In this article, I summarize and illustrate with examples how visualization can help understand different aspects of the pandemic.**

The deadly impact of COVID-19 is driving a massive amount of research that aims at understanding the various characteristics of the pandemic. While there is no vaccine, considerable effort has been devoted to understanding the spread of the disease in different places in the world. The speed with which the disease has spread throughout the world demands agile solutions to understand and estimate the disease progression.

Interactive *dashboards* with several charts surfaced in different formats to offer concise ways to express the pandemic's growth. Figure 1 illustrates some examples. The dashboard developed by Johns Hopkins University (JHU)[1] was the first to track and display information on cases and death totals for different countries and states in the United States. Along with lists of total counts and histograms, a *bubble map* composed of circles of different radii allows a visual inspection of how serious the pandemic is around the world. Interesting plots were created on news outlets such as the New York Times (NYT)[2,3] and the Washington Post.[3] For example, NYT used *Choropleth maps*, a representation where geographical regions (countries or states) are mapped to colors associated with a measurement for that region (e.g., number of cases). This representation is useful to communicate trends, such as the average daily counts for the past week. Similarly, they display the time series evolution for each region using *line heatmaps*, where daily values are mapped to colors and displayed in a row. Our work (described in more detail in the following) used both Choropleth maps and

Digital Object Identifier 10.1109/MCSE.2020.3019834
Date of current version 9 October 2020.

JHU COVID-19 Dashboard https://coronavirus.jhu.edu/map.html (copyright 2020 Johns Hopkins University, all rights reserved)



COVID-19 Dashboard by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University (JHU)

Global Cases
20,124,437

Cases by Country/Region/Sovereignty

5,097,164	US
3,057,470	Brazil
2,268,675	India
895,691	Russia
563,598	South Africa
485,836	Mexico
483,133	Peru
397,623	Colombia
375,044	Chile
331,189	Iran
322,980	Spain
313,393	United Kingdom
291,468	Saudi Arabia
285,191	Pakistan
263,503	Bangladesh
253,868	Argentina
250,825	Italy
241,997	Turkey
239,349	France

Admin0 Admin1 Admin2

Last Updated at (M/D/YYYY)
8/11/2020, 10:35:19 AM

188
countries/regions

Cumulative Cases | Active Cases | Incidence Rate | Case-Fatality Ratio | Testing Rate | Hospitalization Rate

Esri, FAO, NOAA

Lancet Inf Dis Article: Here. Mobile Version: Here. Data sources: Full list. Downloadable database: GitHub, Feature Layer.
Lead by JHU CSSE. Technical Support: Esri Living Atlas team and JHU APL. Financial Support: JHU, NSF, Bloomberg Philanthropies and Stavros Niarchos Foundation. Resource support: Slack, Github and AWS. Click *here* to **donate** to the CSSE dashboard team, and other JHU COVID-19 Research Efforts. FAQ. Read more in this blog. Contact US.

Cases and **Death** counts include confirmed and probable (where reported).
Recovered cases are estimates based on local media reports, and state and local reporting when available, and therefore may be substantially lower than the true number. US

Global Deaths
737,224

163,505 deaths	US
101,752 deaths	Brazil
53,003 deaths	Mexico
46,611 deaths	United Kingdom
45,257 deaths	India
35,209 deaths	Italy
30,327 deaths	France
28,576 deaths	Spain
21,276 deaths	

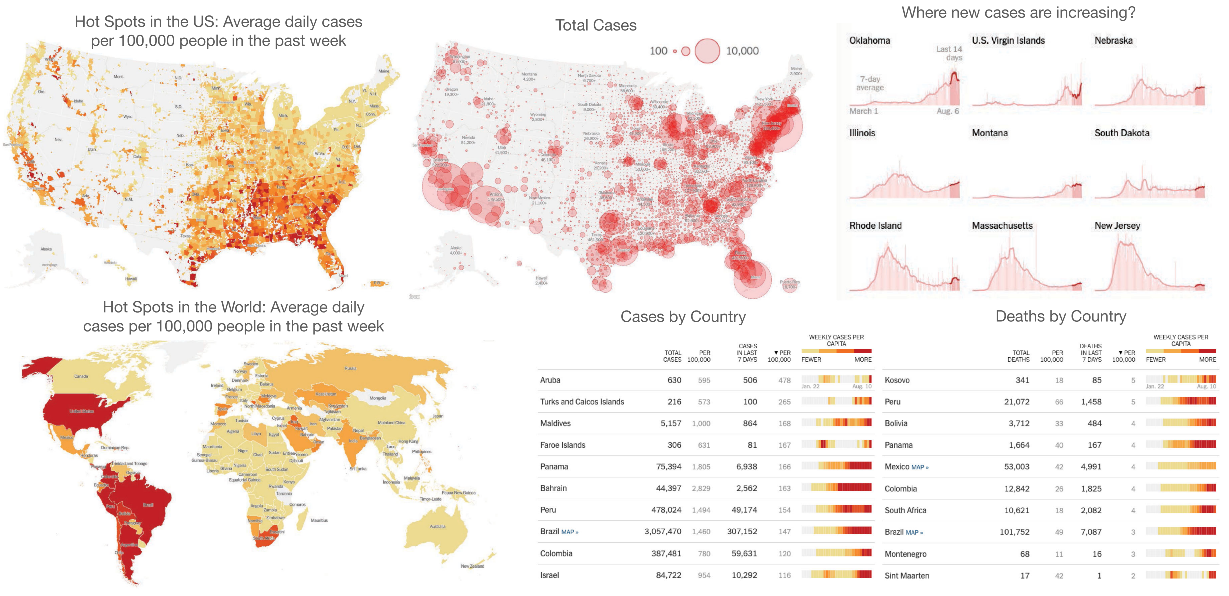
Global Deaths | Global Recovered

US State Level
Deaths, Recovered

32,781 deaths, 73,736 recovered	New York US
15,878 deaths, 33,013 recovered	New Jersey US
10,481 deaths, recovered	California US
9,237 deaths, 349,833 recovered	Texas US
8,741 deaths, 99,021 recovered	Massachusetts US
8,277 deaths, recovered	Florida US
7,846 deaths, recovered	Illinois US
7,314 deaths, 91,978 recovered	Pennsylvania US
6,526 deaths, 63,636 recovered	

US Deaths, Recovered

The New York Times https://www.nytimes.com/interactive/2020/us/ (used with permission)



Hot Spots in the US: Average daily cases per 100,000 people in the past week
Total Cases
100
10,000
Where new cases are increasing?
Oklahoma
Last 14 days
7-day average
March 1
Aug. 6
U.S. Virgin Islands
Nebraska
Illinois
Montana
South Dakota
Rhode Island
Massachusetts
New Jersey
Hot Spots in the World: Average daily cases per 100,000 people in the past week

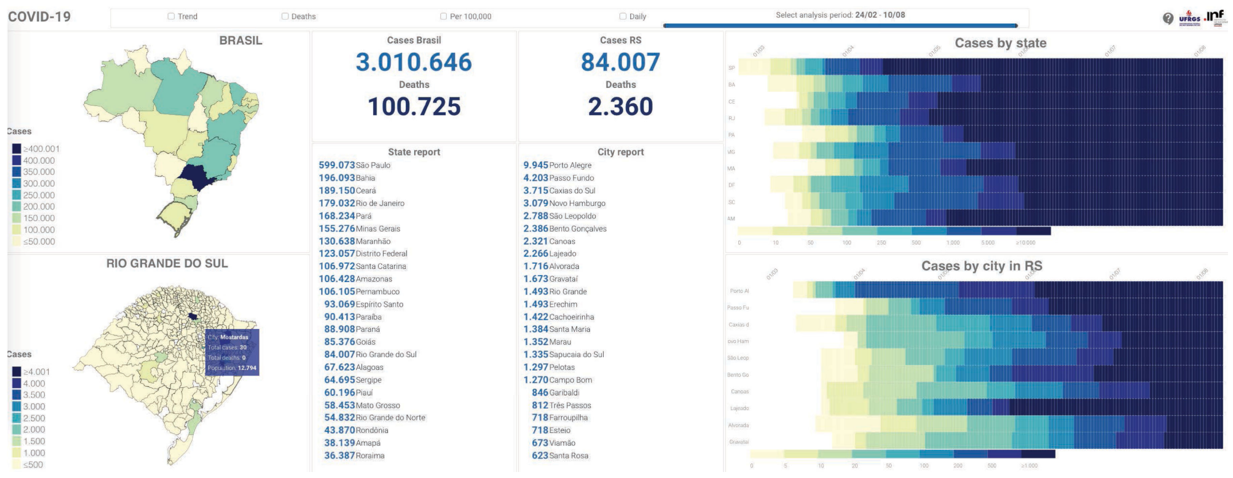
Cases by Country

	TOTAL CASES	PER 100,000	CASES IN LAST 7 DAYS	▼ PER 100,000	WEEKLY CASES PER CAPITA FEWER MORE
Aruba	630	595	506	478	Jan. 22 Aug. 10
Turks and Caicos Islands	216	573	100	265	
Maldives	5,157	1,000	864	168	
Faroe Islands	306	631	81	167	
Panama	75,394	1,805	6,938	166	
Bahrain	44,397	2,829	2,562	163	
Peru	478,024	1,494	49,174	154	
Brazil MAP »	3,057,470	1,460	307,152	147	
Colombia	387,481	780	59,631	120	
Israel	84,722	954	10,292	116	

Deaths by Country

	TOTAL DEATHS	PER 100,000	DEATHS IN LAST 7 DAYS	▼ PER 100,000	WEEKLY CASES PER CAPITA FEWER MORE
Kosovo	341	18	85	5	Jan. 22 Aug. 10
Peru	21,072	66	1,458	5	
Bolivia	3,712	33	484	4	
Panama	1,664	40	167	4	
Mexico MAP »	53,003	42	4,991	4	
Colombia	12,842	26	1,825	4	
South Africa	10,621	18	2,082	4	
Brazil MAP »	101,752	49	7,087	3	
Montenegro	68	11	16	3	
Sint Maarten	17	42	1	2	

Brazil Dashboard https://covid19.ufrgs.dev/dashboard-int



Cases Brasil
3.010.646
Deaths
100.725

Cases RS
84.007
Deaths
2.360

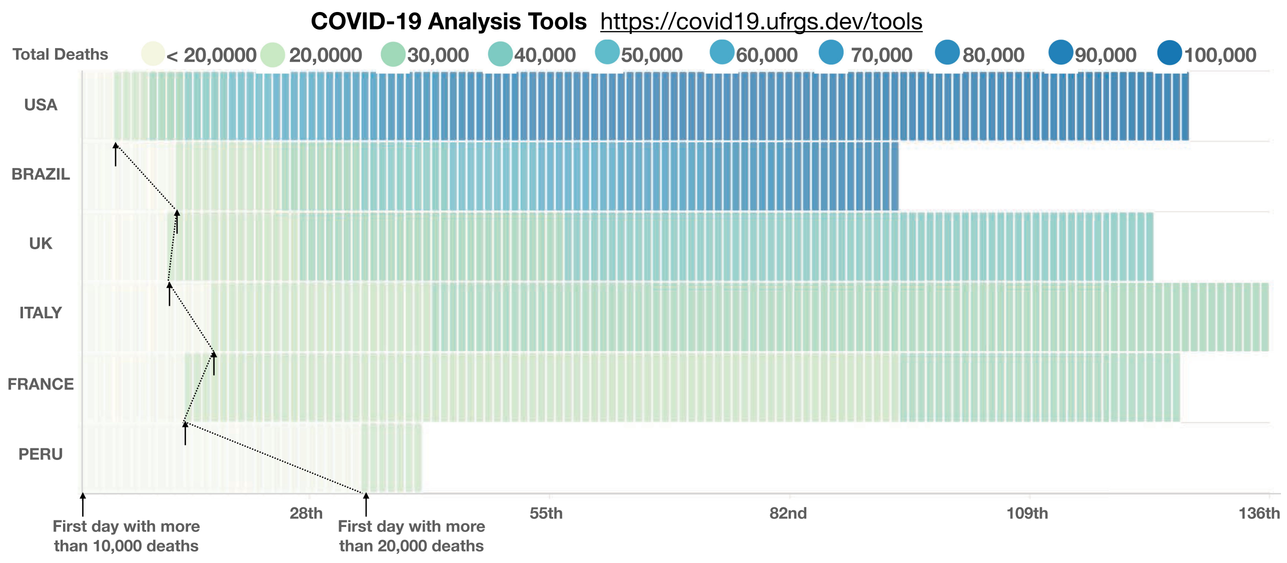
State report

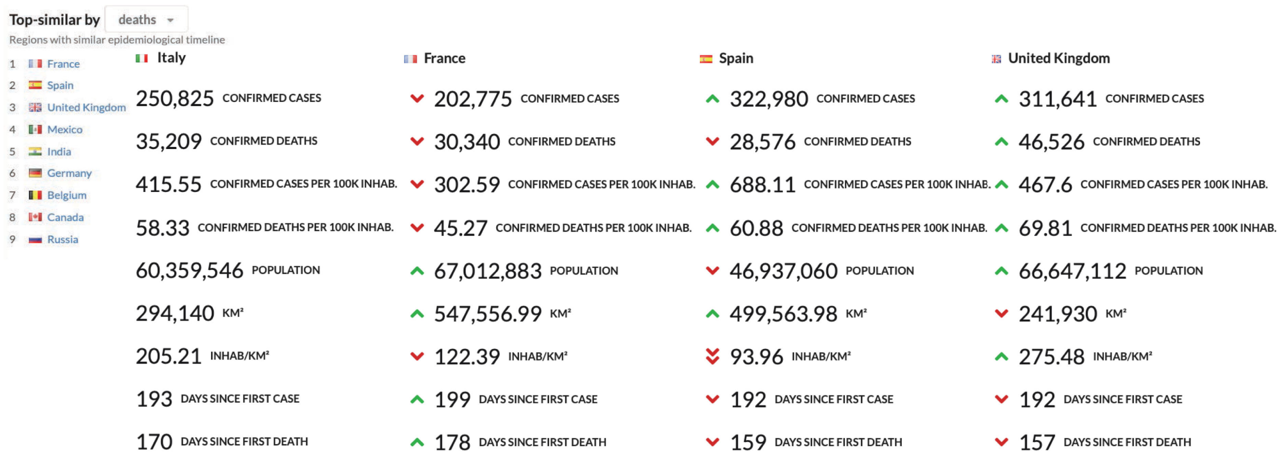
599.073	São Paulo
196.093	Bahia
189.150	Ceará
179.032	Rio de Janeiro
168.234	Pará
155.276	Minas Gerais
130.638	Maranhão
123.057	Distrito Federal
106.972	Santa Catarina
106.428	Amazonas
106.105	Pernambuco
93.069	Espírito Santo
90.413	Paraíba
88.908	Paraná
85.376	Goiás
84.007	Rio Grande do Sul
67.623	Alagoas
64.695	Sergipe
60.196	Piauí
58.453	Mato Grosso
54.832	Rio Grande do Norte
43.870	Rondônia
38.139	Amapá
36.387	Roraima

City report

9.945	Porto Alegre
4.203	Passo Fundo
3.715	Caxias do Sul
3.079	Novo Hamburgo
2.788	São Leopoldo
2.386	Bento Gonçalves
2.321	Canoas
2.266	Lajeado
1.716	Alvorada
1.673	Gravataí
1.493	Rio Grande
1.493	Erechim
1.422	Cachoeirinha
1.384	Santa Maria
1.352	Marau
1.335	Sapucaia do Sul
1.297	Pelotas
1.270	Campo Bom
846	Garibaldi
812	Três Passos
718	Farroupilha
718	Esteio
673	Viamão
623	Santa Rosa

Figure 1. Dashboards and interactive tools to analyze COVID-19 data: Johns Hopkins University Dashboard, The New York Times charts (used with permission), and INF-UFRGS Brazil Dashboard.





Top-similar by deaths

Regions with similar epidemiological timeline

1 France
2 Spain
3 United Kingdom
4 Mexico
5 India
6 Germany
7 Belgium
8 Canada
9 Russia

Italy	France	Spain	United Kingdom
250,825 CONFIRMED CASES	▼ 202,775 CONFIRMED CASES	▲ 322,980 CONFIRMED CASES	▲ 311,641 CONFIRMED CASES
35,209 CONFIRMED DEATHS	▼ 30,340 CONFIRMED DEATHS	▼ 28,576 CONFIRMED DEATHS	▲ 46,526 CONFIRMED DEATHS
415.55 CONFIRMED CASES PER 100K INHAB.	▼ 302.59 CONFIRMED CASES PER 100K INHAB.	▲ 688.11 CONFIRMED CASES PER 100K INHAB.	▲ 467.6 CONFIRMED CASES PER 100K INHAB.
58.33 CONFIRMED DEATHS PER 100K INHAB.	▼ 45.27 CONFIRMED DEATHS PER 100K INHAB.	▲ 60.88 CONFIRMED DEATHS PER 100K INHAB.	▲ 69.81 CONFIRMED DEATHS PER 100K INHAB.
60,359,546 POPULATION	▲ 67,012,883 POPULATION	▼ 46,937,060 POPULATION	▲ 66,647,112 POPULATION
294,140 KM²	▲ 547,556.99 KM²	▲ 499,563.98 KM²	▼ 241,930 KM²
205.21 INHAB/KM²	▼ 122.39 INHAB/KM²	▼▼ 93.96 INHAB/KM²	▲ 275.48 INHAB/KM²
193 DAYS SINCE FIRST CASE	▲ 199 DAYS SINCE FIRST CASE	▼ 192 DAYS SINCE FIRST CASE	▼ 192 DAYS SINCE FIRST CASE
170 DAYS SINCE FIRST DEATH	▲ 178 DAYS SINCE FIRST DEATH	▼ 159 DAYS SINCE FIRST DEATH	▼ 157 DAYS SINCE FIRST DEATH

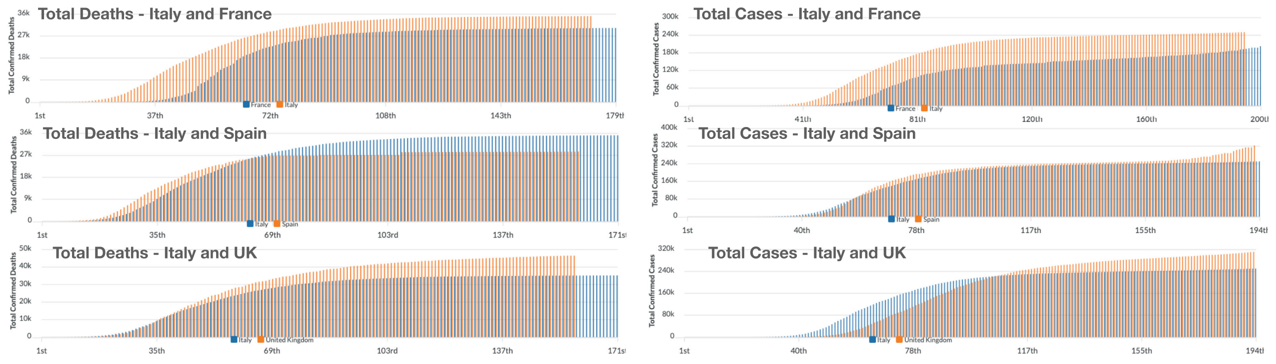
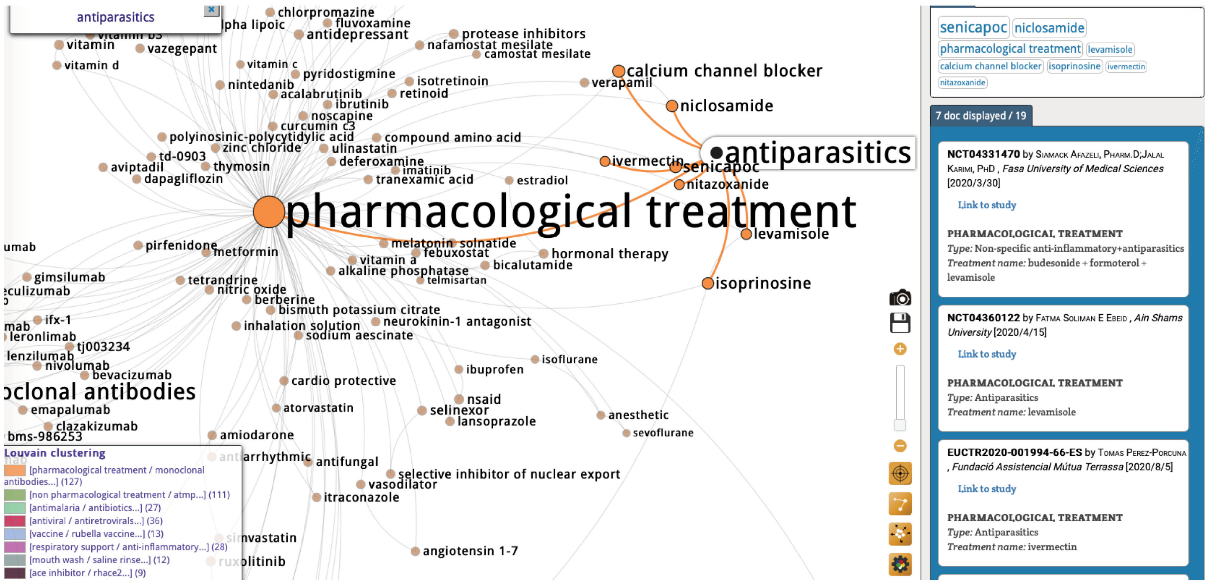


Figure 2. (Top) Heatmap matrices are useful for comparing time series such as the total deaths for different countries. Columns can be aligned by the first date after reaching a certain threshold, which allows us to compare when countries passed through specific checkpoints. (Bottom) Searching for places with similar timelines of deaths to Italy.

Literature Exploration of CoronaVirus Clinical Trials https://covid-nma.com/treatment_mapping/



senicapoc niclosamide pharmacological treatment levamisole calcium channel blocker isoprinosine ivermectin nitazoxanide

7 doc displayed / 19

NCT04331470 by Siamack Afazeli, Pharm.D;Jalal Karimi, PhD , *Fasa University of Medical Sciences* [2020/3/30]

Link to study

PHARMACOLOGICAL TREATMENT
Type: Non-specific anti-inflammatory+antiparasitics
Treatment name: budesonide + formoterol + levamisole

NCT04360122 by Fatma Soliman E Ebeid , *Ain Shams University* [2020/4/15]

Link to study

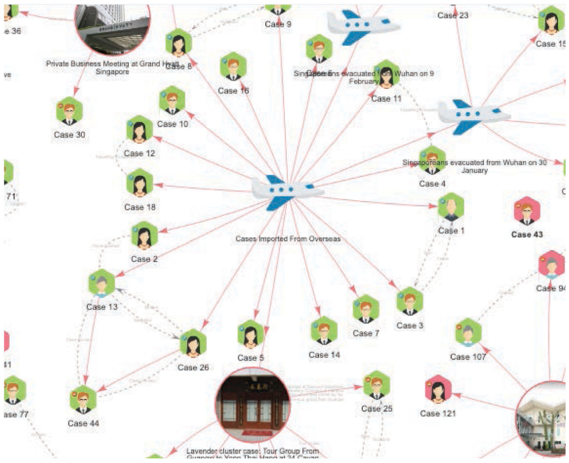
PHARMACOLOGICAL TREATMENT
Type: Antiparasitics
Treatment name: levamisole

EUCTR2020-001994-66-ES by Tomas Perez-Porcuna , *Fundació Assistencial Mútua Terrassa* [2020/8/5]

Link to study

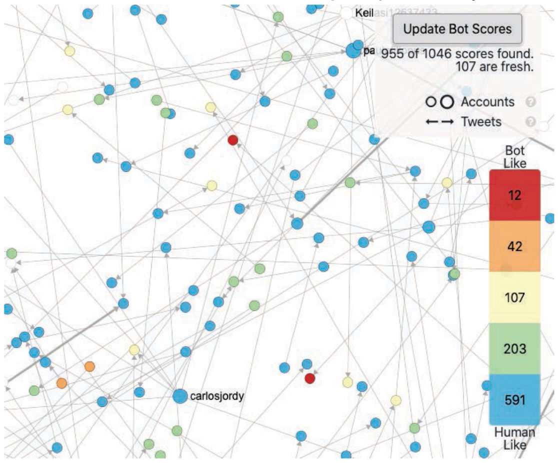
PHARMACOLOGICAL TREATMENT
Type: Antiparasitics
Treatment name: ivermectin

Contact Tracing https://co.vid19.sg/singapore/clusters



Case 9
Case 23
Case 15
Private Business Meeting at Grand Hyatt Singapore
Case 8
Case 16
Singaporeans evacuated from Wuhan on 9 February
Case 11
Case 30
Case 10
Case 12
Singaporeans evacuated from Wuhan on 30 January
Case 4
Case 18
Case 1
Case 43
Cases Imported From Overseas
Case 2
Case 94
Case 13
Case 3
Case 7
Case 26
Case 5
Case 14
Case 107
Case 25
Case 121
Case 77
Case 44
Lavender cluster case: Tour Group From Guangxi to Yong Thai Hang at 24 Cavan

Spread of claims and fact checking https://hoaxy.iuni.iu.edu



Simulation of the transport and spread of Corona virus through the air - Vuorinen et al. [9]

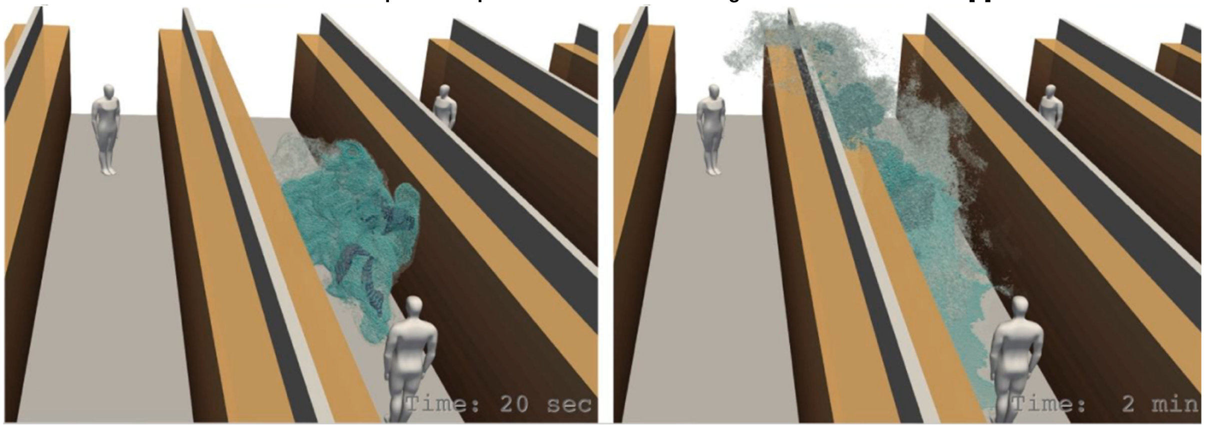


Figure 3. Applications: Literature exploration, contact tracing, spread of claims and fact checking, and simulation of the transport and spread of the novel coronavirus.

line heatmaps in the development of dashboard specially designed for Brazil. Using a focus-plus-context interface with coordinated views, the user can inspect the context using two Choropleth maps (one for the states and a second for a state selected), as well as details using two *matrix heatmaps* (collection of stacked line heatmaps) for states and cities. The tool displays daily or cumulative data, absolute or relative to population, and supports filtering by a time interval.

A vast collection of community-developed dashboards and interactive tools about COVID-19 are available. Good starting places to look are the data hub hosted by Tableau and the top 100 R-resources organized by Soetewey.[4] In-depth analysis is available at sites, such as *Our World in Data*,[5] *Bing*,[6] and the *COVID Tracking Project*,[7] among others. After developing the Brazilian dashboard, we devoted our efforts to create a set of tools to compare the spread of COVID-19 data in different regions of the world.[8] We collected data from more than 6 000 locations in the world, and our interface has different charts that support visualizing multiple locations in a single chart. Since the pandemic is at different stages in the world, we allow the user to align the time-series of data by a certain data the series passes a given threshold (e.g., after 100 cases). This representation is useful to observe when different locations passed through specific checkpoints (top of Figure 2).

While our initial charts support the comparison of different regions, the tool required the user to drive the process of choosing the regions to compare. From the beginning, we felt the need to have an automatic tool that could, given a region of interest, return the closest regions given a similarity function. Looking at the disease's spread in distinct places, but with similar growth patterns, can be useful to predict behaviors. Thus, we developed a search engine to support queries using different similarity functions. For example, at the bottom of Figure 2, we show the results of searching for regions similar to Italy concerning the number of deaths. The results are listed in a ranking, with pairwise comparisons that detail attributes of the different locations and evolution charts, aligned by the day of the first death. We observe the similarities in the evolution in the number of deaths and cases for Italy, France, Spain, and the United Kingdom. While using the tool, we also saw similarities among cities from Brazil and the United States, both countries with large COVID-19 numbers.

The examples so far give a glimpse of how data visualization can help in the understanding of COVID-19. Figure 3 illustrates other applications where data visualization can help. The first example shows how *multidimensional projections* and *network visualizations* can help the *literature exploration* of papers that describe novel coronavirus research. As new research about COVID-19 is published, there is a great need to review up-to-date literature and treatments conveniently. *Contact tracing* is another application that relies on graph visualization to trace the network of people who may have been in contact with a COVID-19 patient, an activity essential to control the dissemination of the disease and essential for directing social distancing regulations. Graph visualization is also important in for *social media spread and fact checking*. With many people at home, social networks are playing a significant role in people's lives these days. Unfortunately, the dissemination of fake news and automated posting from robots is also rising. Fact-checking over the propagation network can help identify misleading information and patterns of dissemination. The fourth and final example highlights the importance of data visualization in the analysis of scientific simulations. It shows the visualization of simulating the transport and spread of novel coronavirus in closed spaces,[9] which shows how an infected person can disseminate the virus indoors.

Many other examples include data visualization in the analysis of COVID-19 data, and many more is surfacing every day. We hope this summary highlights interesting examples, give pointers to other references, and motivate people to pursue other applications.

REFERENCES

1. E. Dong, H. Du, and L. Gardner, "An interactive web-based dashboard to track COVID-19 in real time," *Lancet Infectious Diseases*, vol. 20, pp. 533–534, 2020, doi: 10.1016/S1473-3099(20)30120-1.
2. The New York Times, "Coronavirus map: Tracking the global outbreak," 2020. [Online]. Available: https://www.nytimes.com/interactive/2020/world/coronavirus-maps.html

3. The Washington Post, “Mapping the worldwide spread of the coronavirus,” 2020. [Online]. Available: https://www.washingtonpost.com/graphics/2020/world/mapping-spread-new-c oronavirus/
4. A. Soetewey, “Top 100 R resources on novel COVID-19 coronavirus,” 2020. [Online]. Available: https://www.statsandr.com/blog/top-r-resources-on-covid-19-coronavirus
5. D. Kennedy, A. Seale, D. Bausch, H. Ritchie, and M. Roser, “How experts use data to identify emerging COVID-19 success stories,” 2020. [Online]. Available: https://ourworldindata.org/identify-covid-exemplars
6. Microsoft BING, “COVID-19 live map tracker by Microsoft,” 2020. [Online]. Available: https://www.bing.com/covid/
7. The Atlantics, “COVID tracking project,” 2020. [Online]. Available: https://covidtracking.com
8. F. Pontes, F. Pinto, W. Leuschner, and J. L. D. Comba, “COVID-19 analysis tools at Instituto de Informática-ufrgs,” 2020. [Online]. Available: https://covid19.ufrgs.dev
9. V. Vuorinen *et al.*, “Modelling aerosol transport and virus exposure with numerical simulations in relation to SARS-CoV-2 transmission by inhalation indoors,” *Safety Sci.*, vol. 130, 2020, Art. no. 104866. doi: 10.1016/j.ssci.2020.104866.

João L. D. Comba is currently a full professor in computer science with Instituto de Informática—UFRGS, Porto Alegre, Brazil. His research interests include visualization, visual analytics, data science, geometric algorithms, and spatial data structures. He received the Ph.D. degree in computer science from Stanford University, Stanford, CA, USA. Contact him at comba@inf.ufrgs.br.

Lattice Gas Cellular Automata Fluid Dynamics Case Study

Micah D. Schuster
Wentworth Institute of Technology

***Abstract*—The Navier–Stokes equations are the basis for describing the flow of a viscus material and are used to model fluid motion from weather to air flow over a wing. These equations, however, tend to be notoriously difficult to solve, whether analytically or computationally, even for small systems due to their highly nonlinear nature. Thus, less complicated methods that are more computationally tractable are desirable in domains as varied as hydroelectric power to race car construction. At a fundamental level, fluids are composed of interacting molecules. Lattice Gas Cellular Automata (LGCA) represents an efficient way to simulate these interacting fluid particles on a lattice. LGCA captures the microscopic behavior of the fluid by applying simple collision and propagation rules at each lattice site. This leads to realistic macroscopic behavior that can be used to build insight about real fluid flow. Here, we show the Hardy, Pomeau, and de Pazzis model for simulating a lattice gas. The computational framework presented in this case study can also be expanded to more complicated LGCA.**

Cellular automata (CA) is a discrete model often used in computer science, mathematics, physical sciences, and biological sciences to model complex behavior starting with a simple set of rules. The concept was originally developed by Stanislaw Ulam and John von Neumann in the 1940s at Los Alamos National Laboratory for work on crystal growth and self-replicating robots. Further work was done by scientists and mathematicians throughout the rest of the century. Stephen Wolfram, in the 1980s, explored CA in the context of mathematics, physics, biology, and chemistry. His book on this research, *A New Kind of Science*,[1] was published in 2002 and is an important starting point for learning about physical modeling using CA. This case study focuses on a simple fluid model that is easily implemented yet retains the

Digital Object Identifier 10.1109/MCSE.2020.3019164
Date of current version 9 October 2020.

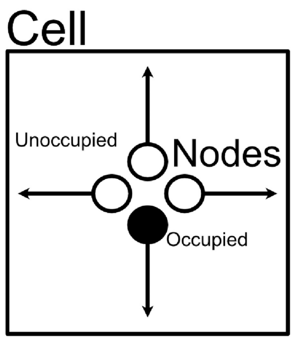


Figure 1. Each grid cell contains four nodes, representing the cardinal directions.

macroscopic fluid behavior present in more complicated simulations.

In the early 1970s, Hardy, Pomeau, and de Pazzis[2] introduced the first CA fluid dynamics simulation. The model is known as the HPP model after the last names of the authors. It was the first of a class of CA that incorporated point particles that move along a lattice. Thus, the state of each cell represents particle interactions.

The lattice is composed of two parts: cells and nodes. The cells are the traditional representation of the square lattice. Each cell, however, contains four nodes, one for each cardinal direction. Thus, the state of each cell is determined by the specific occupation of the nodes, see Figure 1. The nodes themselves represent the fluid particles entering or exiting each cell. This sublattice of nodes is tightly coupled to the amount of data that each cell must contain to represent its current state. For example, if all four nodes contain particles, one could use the value "4" to represent that state. This representation will be the key to performing the simulation update in a concise manner.

Computing the next state of the system is significantly different than traditional CA. The HPP model updates in two distinct steps: particle propagation from one lattice site to another and intracell particle collision. Each is computed separately on any given iteration. The discretization of these dynamics allows us to define a set of rules that approximate fluid-like behavior.

SETUP

The initial setup of the HPP simulation requires two choices by the user: the grid dimensions and the node representation. A grid size of 512 or 1024 in each dimension will be sufficient for our test application. The choice of grid size is important for two main reasons: to reduce the noise that arises because of the granular nature of the simulation by averaging cell occupation, defined by the number of nonzero nodes in a cell, and to increase the size of the domain to add more complex features like obstacles.

Since each of the four nodes within the cells can be occupied or unoccupied at any given iteration of the simulation, they only require a single bit to represent their occupation. Thus, using a single byte for each cell gives ample memory for large grids. Most languages have a data type that represents a single byte, e.g., `unsigned char` in C/C++ or `uint8` in MATLAB. We will use the convention that the first four bits represent up-left-right-down. For example, a cell with occupied up and down nodes would be `1001`.

Any CA uses two buffers, which are two identical grids. One contains the current state of the simulation and is read from to apply our simulation rules. The second buffer is written to as the current iteration updates the lattice. Without this double buffer approach we would obtain incorrect results after the update.

Typically, we choose a random initial occupation for each cell, between 0 and 4, to represent the average density of particles in the domain. To make the simulation more interesting, we will add a high-density region, i.e., an occupation of four for all cells within a given region. This will create familiar wave-like behavior within the domain.

PROPAGATION

Each individual particle in the simulation is represented by an occupied node. The rules by which these particles move from cell to cell are important for understanding the propagation step.

- Each particle of the fluid moves in one of the four cardinal directions to a new cell determined by the node that it occupies at the propagation step.
- The particles all move at the same speeds, chosen to be one for convenience.
- The particles never move diagonally in the HPP model.

There are compact ways to represent the propagation using a series of bitwise operations, however, for clarity, we will present the algorithm using a series of masks and bit shifts.

We will also include a getter function for each direction that gets the index of that adjacent cell. The appropriate mask is then applied to that cell, which isolates a specific node. For example, using the hex value `0×01` will isolate a particle that will move into the current cell from above. Most languages need a `0x` or `0b` in front of the number to represent hexadecimal or binary, respectively. We then bit shift that value into the correct position for the current cell. In effect, an occupied bottom node in the above cell will propagate into the top node of the current cell. We then do this process for each direction to build the new current cell. This process is shown in the Listing 1.

Listing 1. Propagate function.

```
void propagate(){
 unsigned char up, down, left, right, final;

 for(int i = 0; i < total_size; i++){
  up = (buffer1[GetUp(i)] & 0x01) << 3;
  left = (buffer1[GetLeft(i)] & 0x02) << 1;
  right=(buffer1[GetRight(i)] & 0x04) >> 1;
  down = (buffer1[GetDown(i)] & 0x08) >> 3;
  final = up | down | left | right;
  buffer2[i] = final;
 }
  swapBuffers(&buffer1, &buffer2);
}
```

ACTIVITY 1: Basic Propagation With Bitwise Operators

When we propagate particles from cell to cell, we use a bit mask and a bit shift. Each mask is simply a number that we apply to a grid location using a bitwise `AND` operator. The way we represent the number does not actually matter but is easier to visualize when using binary or hexadecimal. Test this operation by starting with the value 9, or `0b1001` when using a binary number in your code.

- Create a variable with a value of 1, or `0b0001` in binary, to use as our mask.
- Apply the bitwise `AND` (`&` in most languages): `9 & 1`. In binary form, this operation is `0b1001 & 0b0001 = 0b0001`.
- Now apply a bit shift to the result, for example, `0b0001 << 3`. This will shift all the bits to the left by three spaces. Verify the result to be 8, `0b1000` in binary.

Determine all the masks that we need for each of the four cardinal directions.

Here, `total_size`, `buffer1`, and `buffer2` can either be global or passed into the function. We loop over every cell in the simulation, `total_size`, in this case using a one-dimensional array. The two buffers are arrays that represent the two grids that we work with. `buffer1` will always contain the current state of the grid and `buffer2` is the grid that is written to during the propagation step. After updating the grid, the buffers are swapped to prepare for the next propagation step. The four getter functions allow us to get the appropriate grid index for the current cell's neighbor.

ACTIVITY 2: Propagation at the Boundary

Attempting to access elements beyond the edge of the grid in the above code will cause major problems when running the simulation.

- Try to think of several options for handling the boundary cells.
- In this case, we will consider the first and last row/column to be fixed, i.e., no propagation occurs when particles enter these boundary nodes. Effectively particles "bounce" off the walls. Implement this condition.

COLLISION

The calculation of the collision step is straightforward in that it only requires knowledge of the current occupation of the cell and the possible collisions that the HPP model allows.[2] In this step, we assume that each occupied node represents particles moving into the cell. Once the collisions have been applied, the nodes will represent the particles moving in the outgoing direction, which prepares us for the next propagation step.

The most important requirement for proper collisions is the conservation of momentum. In the HPP model, each particle is assumed to have the same mass and the same speed when moving along the lattice. Thus, the momentum,

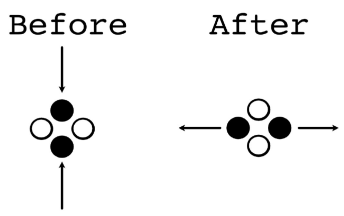


Figure 2. Rotationally symmetric two particle collision.

the product of mass, and velocity, can be determined solely by the direction of motion. Due to the square lattice, this requirement restricts the possible types of collisions to two (though they are the same collision, just rotated 90°). Figure 2 shows the cell state before and after collision. In both configurations, the total momentum in the cell is zero because the particles are moving in opposite directions, thus the collision conserves momentum. All other cell states update as if no collision occurs, i.e., the particles simply move through each other.

The code in Listing 2 shows the collide function. We obtain a reference to each cell, invoke the lookup table by sending the current state as the index and getting back the state resulting from the collision, and save the result to `buffer2`. The buffers are then swapped, similar to Listing 1. Again, `total_size`, `buffer1`, and `buffer2` can either be global or passed into the function.

Listing 2. Collide function.

```
void collide(){

  for(int i = 0; i < total_size; i++){
    unsigned char node = buffer1[i];
    buffer2[i] = collisionLookup[node];
  }

  swapBuffers(&buffer1, &buffer2);
}
```

ACTIVITY 3: Basic Collisions

Our collision code will use a lookup array where the index of the array represents the state of the cell before collision and the value is the state after the collision. For example, index 1, binary value `0b0001`, represents a particle in the down position. After the collision step, that particle will be in the up position because it passes through the cell with no other particles to interact with. The value of the array at index 1 will then be 8, or `0b1000`.

Adding obstacles in the domain can be done by utilizing an additional bit in the data type representing the cell. One simply checks if the bit is set. If it is, the collision is not performed. This effectively causes the particle to bounce off the obstacle.

VISUALIZATION

CA fluid dynamics models tend to be very noisy. This is due to the visualization of the occupation at each lattice point via a color. This represents the microscopic structure of our fluid, which while interesting, is not as useful for viewing large scale, macroscopic behaviors.

The first step to create our macroscopic visualization is to consider a second grid that contains fewer cells to display. For example, if the microscopic lattice is 1024×1024, we can create a smaller lattice of dimension 64×64. To generate the value for each cell in the smaller grid, we will sum the particle occupation in each 16×16 region on our large lattice. Then, we divide by 1024, representing the total possible occupation for the 16×16 region (256 total cells with 4 possible particles per cell). Effectively, we are computing the spatial average over sets of lattice points, which is the average particle density.

Choosing the appropriate size for the microscopic lattice and the associated averaging region depends largely on the structure of the domain that you want to simulate. The general rule is that if L is size of the macroscopic domain, l_a is the size of the averaging region, and a is the spacing between lattice points in the microscopic lattice. One should choose these values such that

$$L \gg l_a \gg a. \tag{1}$$

For CA fluid simulations the particle density is often averaged over 32×32 or 64×64 lattice points. We can also include momentum as part of our visualization. As we mentioned before, because of our assumptions for mass and velocity of particles, the momentum in a specific dimension is either -1 or 1, depending on the direction of motion of the particle. Thus, the average momentum on the macroscopic grid can be computed by summing the individual particle momenta. This

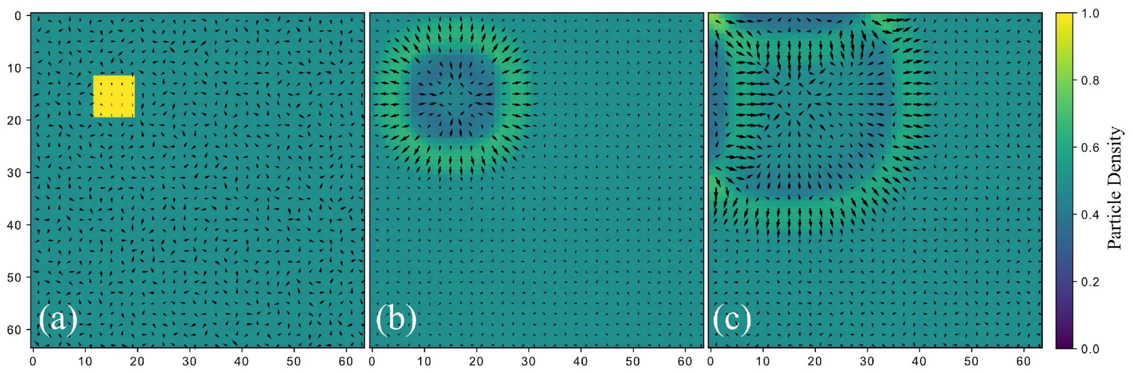


Figure 3. Evolution of a 2048×2048 HPP simulation at (a) the initial condition, (b) after 500, and (c) after 1000 iterations. The CA lattice is averaged over 32×32 cells to obtain particle density, shown by the colorbar, and momentum vectors, shows as arrows.

can be visualized by displaying an arrow for each macroscopic grid location.

Figure 3 combines all the above visualizations and shows the evolution over time of the average density and average momentum. We use a microscopic lattice of 2048×2048 with a macroscopic lattice of 32×32. We can now see the average momentum as the wave propagates through the domain. The initial condition shows a uniform particle density, except for the high-density region, and random momentum vectors which correspond with the underlying random starting lattice.

CONCLUSION

In this case study, we have learned how to represent, calculate, and visualize fluid particles, their propagation, and collisions over time. After mastering the HPP model, we recommend attempting to implement the FHP model,[3] developed by Frisch, Hasslacher, and Pomeau in 1986. This fluid model lies on a hexagonal lattice, thus increasing the number of possible interactions at a given lattice location. Overall, this creates a more realistic fluid simulation.

ACKNOWLEDGMENTS

This work was performed with the support of the Department of Computer Science and Networking at Wentworth Institute of Technology.

REFERENCES

1. S. Wolfram, *A New Kind of Science*. Champaign, IL, USA: Wolfram Media, 2002.
2. J. Hardy, O. de Pazzis, and Y. Pomeau, “Molecular dynamics of a classical lattice-gas: Transport properties and time correlation functions,” *Phys. Rev. A*, vol. 15, no. 5, pp. 1949–1961, 1976.
3. U. Frisch, B. Hasslacher, and Y. Pomeau, “Lattice-gas automata for the Navier–Stokes equation,” *Phys. Rev. Lett.*, vol. 56, no. 14, pp. 1505–1508, 1986.

Micah D. Schuster is currently an Assistant Professor with the Department of Computer Science and Networking, Wentworth Institute of Technology, Boston, Boston, MA, USA. He received the Ph.D. degree in computational science from San Diego State University, San Diego, CA, USA. Contact him at schusterm@wit.edu.

Diversity and Inclusion Through Leadership During Challenging Times

Mary Ann Leung
Research, Sustainable Horizons Institute

By some measures, significant progress has been made in diversifying computing in science and engineering (CSE). The first Grace Hopper Celebration of Women in Computing conference brought together 500 technical women in 1994, while the 2020 conference is expecting 30 000 attendees. Diversity organizations in CSE exist in a variety of diversity dimensions such as gender, race, ethnicity, sexual orientation, and technical areas such as computing, math, and engineering. For example, we have organizations focused on women in mathematics, Blacks in engineering, LGBT in math, and Hispanics in computing, as well as many other combinations. The growth of these organizations represents important *progress* and *maturity* toward diversifying. This progress was accomplished by a focus on the "D" in diversity and inclusion. That is, a focus on the underrepresented by providing important tools, community, and resources; or "fixing the minorities."

Yet, the data show considerable gaps still remain between the available talent and actual participation by these groups, as well as higher dropout rates for underrepresented groups. Further progress calls for new approaches to diversity and inclusion. The maturity of the "D" suggests addressing inclusion, or the "I" side. This requires not just a focus on marginalized people, but also efforts to transform the broader communities to places where *all* people are welcome, productive, and thriving; environments where inclusion is the norm,[1] i.e., normal practice.

Recent events, including the COVID-19 pandemic and the Black Lives Matter movement, highlight longstanding systemic issues related to racial and ethnic disparities. This spotlight created a global focus on diversity and a great deal action including protests, new or renewed commitment to diversity, and interest in ways to create change. The momentum generated begs for action. This article describes ways to harness the momentum.

Digital Object Identifier 10.1109/MCSE.2020.3020445
Date of publication 2 October 2020; date of current version 9 October 2020.

WHY TACKLE DIVERSITY AND INCLUSION WITH LEADERSHIP?

Leading in science requires not just coordination of larger and larger teams,[2] but also the ability to spur creativity and innovation, as described in a science leadership handbook,[3] "It is said that the truth shall set us free; yet we need freedom to discover the truth. Thus, leaders in science and technology (S&T) must accept responsibility for the results of their work and for the means they use to accomplish it. Fundamental to that responsibility is respect for facts, for creativity, and for colleagues." Cultivating respect for colleagues from underrepresented groups has proven elusive given the common sources of attrition, e.g., bias, feelings of isolation, and lack of role models, confidence, and social capital.

Considerable effort over the past several decades by policy makers, researchers, and practitioners has resulted in steady but slow progress toward diversifying the S&T workforce. Important outcome includes the creation and significant growth in organizations such as the Association of Women in Computing, National Society of Blacks in Computing, Hispanics in Computing, Black Girls Code, and the Association for LGBT Mathematicians (Spectra). However, CSE has seen very little progress. According to the National Science Foundation Science and Engineering Indicators, representation of women earning doctorates in computing has grown less than 4% from 2000 to 2015. For others earning doctorates, representation of Hispanics in computing has grown under 2%, Blacks or African Americans have almost doubled, but are still under 7%. In mathematics and statistics, the growth is slower, and some declines are seen, as shown in Table 1.

Organizations focused on specific dimensions of diversity create communities where the underrepresented are the majority and provide important resources and voices to the communities they serve. However, changes in representation remains extremely slow. Common approaches focus on increasing the pipeline. This approach has been hindered by higher dropout rates, also known as the "Leaky Pipeline" problem. Achieving significant growth requires efforts in both diversity and inclusion. Inclusion requires transformation of the broader community to prevent higher attrition rates. Such organizational change requires commitment from the leaders who control organizational power structures; it requires *leadership*.

Table 1. 2000 and 2015 Earned Doctorate Degrees at U.S. Academic Institutions (Source: 2018 National Science Foundation Science & Engineering Indicators).

Earned Doctorate at U.S. Institutions	2000 (%)	2015 (%)
COMPUTING		
- Female	18.2	21.9
- Hispanic or Latinx	3.2	4.9
- American Indian or Alaska Native	0.0	0.5
- Black or African American	3.7	6.6
MATH AND STATISTICS		
- Female	28.0	25.2
- Hispanic or Latinoa	2.04	3.89
- American Indian or Alaska Native	0.34	0.11
- Black or African American	2.21	2.22

WHY NOW?

The world changed in early 2020 when the COVID-19 pandemic spread across the globe. Economies, socialization, education, and infrastructure have been dramatically affected as people found new ways of working and living while sheltering in place. The impacts are not just dramatic; for many, they are surreal. In early April, U.S. data started to emerge indicating disproportionate effects on Blacks, Hispanics, and Native Americans. The U.S. National Institutes of Health report a consistent pattern of racial/ethnic differences in the infection and morbidity rates among African American/Black, Latino, American Indian, Alaskan Native, and Pacific Islander populations.[4] For example, comparing hospitalized and nonhospitalized COVID-19 patients at six acute care hospitals in Atlanta, GA, and selected outpatient clinics, blacks represent 79% of hospitalized patients[5] although they are just 52% of the population there. The pandemic also dramatically affects education as most schools have transitioned to online learning. The rapid movement to remote education has created different burdens and, in some cases, huge barriers. The digital divide, parents juggling work from home and new schooling needs and being poorly prepared to serve as teachers, online discrimination and bullying, and systemic racial bias are furthering the pandemic effects on education.[6] It is unclear what long-term affects the pandemic will have on education; however, it is likely that it will exacerbate existing inequities such as lower entry and participation rates and higher dropout rates for minorities.

The disparate effects of the pandemic are intensified by recent events surrounding the Black Lives Matter movement. This combination created a spotlight on issues around race, ethnicity, diversity, and inclusion. In S&T education generally, and specifically in CSE, long standing disparities in representation are emphasized by this spotlight. As a result, organizations and individuals have increased commitment to diversity, similarly to what happened six years ago. In 2014, major technology companies announced commitment to diversity and invested millions of dollars, however little change in metrics occurred. New approaches are needed to capitalize on the current momentum.

DIVERSITY AND INCLUSION LEADERSHIP: CRITICAL MASS AND INTEGRATION

Leadership in science comes with responsibility, as described above; “Fundamental to that responsibility is respect for facts, for creativity, and for colleagues.” Leaders need to develop respect for colleagues. However, the data and the reasons why the numbers are so low, such as bias and feelings of isolation, suggest that this has continued to be a challenge. To form productive teams of people from different backgrounds, leaders need to assemble groups ensuring all team members develop a sense of belonging and respect for differences and each other, while cultivating creativity and collaboration. The current participation by women and minorities in CSE suggests scientists from inhomogeneous backgrounds, in many cases, are the only ones or one of just a few in otherwise majority groups; that is the only female, Black, Hispanic, Native American, etc. The small representation leads to feelings of isolation and low self-efficacy and/or lack of belonging. To combat this situation, critical mass, that is larger groups of underrepresented individuals, is needed, thereby reducing the sense of “I’m the only one.” Diversity focused organizations have demonstrated success with this approach. However, returning to environments where underrepresentation is dominant continues to be problematic. Also essential is the balance of subgroups where minority members achieve critical mass, with integration into the overall group through activities that promote respect. Techniques for how to achieve this balance are described below, beginning with an analogy.

Nature has ingeniously found ways to create complex structures such as leaves and snowflakes. These structures are formed through a natural self-assembly process known as dendritic growth by which individual atoms or molecules form intricate structures. Building on natural self-assembly, scientists have devised techniques to create structures such as carbon nanotubes that have proven extremely valuable for medicine, materials science, and other areas and the resultant generative field of nanotechnology. The distributed flight array of modular robots capable of three-dimensional movement, flight, and response to stimuli[7] is an engineering example of how coordination of modular components produces structures of great use, see Figure 1. Both examples rely on adaptation of naturally occurring phenomena or *coordinated self-assembly* to create larger more useful superstructures.

What if this principle could be applied to leadership in science to promote diversity and inclusion?

Self-assembly of snowflakes and carbon nanotubes exploits natural phenomena. Inventors of nanotubes and robotic flight arrays adjust natural processes by combining individual components into coordinated superstructures. How could this idea be applied to science leadership? Science requires innovation *and* coordination of research and researchers; taking individual units and combining them into larger more productive structures. Creativity requires freedom and respect and research indicates that diversity results in greater innovation.[8] Therefore, leaders must focus on cultivating natural individual capabilities while facilitating cooperation *and* respect in diverse groups. The Agile leadership style, an example of this principle widely used in computing, creates conditions for self-organization where teams collaborate, learn from each other, and get quick feedback. Adapting this approach to support diversity and inclusion requires leadership that promotes the following:

- freedom from bias, isolation, low self-efficacy;

Figure 1. Snowflake, leaf, carbon nanotube, and distributed flight array robot.

- respect for each person's background, differences, needs;
- integration within broad, not segregated communities;
- balance of critical mass with integration.

Example

Like many professional societies, the Society of Industrial and Applied Mathematics (SIAM) is composed predominantly of white males. Guided Affinity Groups at SIAM CSE conferences aim to promote diversity and inclusion by creating small learning groups led by SIAM CSE society members. The groups invert the ratios such that the minority becomes the majority. Each morning, groups with concentrations of women and minorities meet with their leader to discuss technical topics and equip them with knowledge and tools to attend conference events where they are again in the minority, flipping the ratios back. This inversion is repeated throughout the conference, creating a series of inversions illustrated in Figure 2. Feedback from participants indicates the activity effectively instills confidence, increases feelings of self-efficacy and belonging, and teaches society members about available talent from sources they would have never considered or been exposed to otherwise.[9] The program promotes respect and creativity while balancing critical mass and integration.

DISCUSSION AND CALL TO ACTION

The participation of women and minorities in CSE remains low. Considerable progress toward diversifying CSE and more broadly S&T is seen through the creation and growth in organizations focused on supporting members of underrepresented groups, yet the progress remains slow. The coupling of this phenomena with the leaky pipeline problem suggests new approaches are needed. The country continues to endure effects of the pandemic and racial tensions. Rather than trying to get back to normal, the country needs to create new norms through

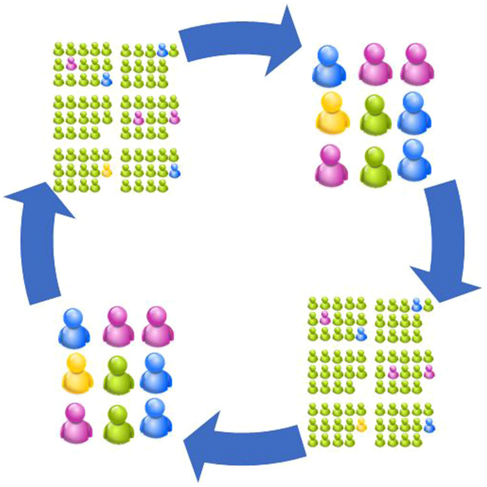


Figure 2. Series of inversions to balance critical mass with integration.

leadership addressing sources of disparities. Leaders are responsible for not only fostering innovation but, also for cultivating respect for facts, creativity, and for colleagues. In the current workforce, women and minorities do not experience equal access or the necessary respect to thrive and innovate. Leaders need to ensure their actions result in 1) freedom through identity; 2) respect for each person's background, differences, and needs; 3) community integration, not segregation; and 4) balance of critical mass with integration. Recent events have highlighted long standing issues, stimulated momentum, and created opportunity to make significant progress.

REFERENCES

1. M. A. Leung, "Developing sustainable methods for broadening participation: transforming mainstream science and technology communities through normalization of inclusion," *Amer. Behav. Scientist*, vol. 62, pp. 683–691, 2018.
2. Committee on the Science of Team Science. *Enhancing the Effectiveness of Team Science*. Washington, DC, USA: National Academies Press, 2015.
3. W. S. Bainbridge, *Leadership in Science and Technology: A Reference Handbook*. Thousand Oaks, CA, USA: Sage Publishing, 2012.
4. E. Perez-Stable, "COVID-19 and racial/ethnic disparities," *J. Amer. Med. Assoc.*, vol. 323, no. 24, pp. 2466–2467, 2020.
5. M. E. Killerby *et.al.*, "Characteristics associated with hospitalization among patients with COVID-19 - Metropolitan Atlanta, Georgia, March–April 2020," *Morbidity Mortality Weekly Rep.*, vol. 69, no. 25, 2020.
6. D. Lederman, "Unequal access to learning, a fall without students and another MOOC movement?" Inside Higher Ed, Apr. 2020.
7. R. Oung, "The distributed flight array: Modular robots that self-assemble, coordinate and take flight," Jun. 12, 2013. Online]. Available: https://robohub.org/the-distributed-flight-array-modular-robots-that-self-assemble-coordinate-and-take-flight/
8. P. Gompers and S. Kovvali, "The other diversity dividend," *Harvard Bus. Rev.*, Jul./Aug. 2018.
9. M. A. Leung, "Broadening engagement by extending the classroom to conferences," in *Proc. Soc. Ind. Appl. Math. Educ. Conf.*, Portland, OR, USA, 2018.
10. M. A. Leung, S. Crivelli, and D. Brown, "Sustainable research pathways: building connections across communities to diversify the national laboratory workforce," in *Collaborative Network for Engineering and Computing Diversity (CoNECD)*, Washington, DC, USA, 2019.

Mary Ann Leung is the founder and president of the Sustainable Horizons Institute, a 501(c) 3 non-profit organization dedicated to developing the scientific workforce with a special interest in creating diverse and inclusive environments. She received the Ph.D. degree in computational physical chemistry from the University of Washington. Contact her at mleung@shinstitute.org.

Editors: Sharon Broude Geva, sgeva@umich.edu
Dirk Colbry, colbrydi@msu.edu

Exemplifying Computational Thinking Scenarios in the Age of COVID-19: Examining the Pandemic's Effects in a Project-Based MOOC

Juan D. Pinto
University of Illinois at Urbana-Champaign,
University of Michigan

Chris Quintana and Rebecca M. Quintana
University of Michigan

***Abstract*—The rapid, global spread of COVID-19 has led to an unprecedented rise in enrollments in online learning experiences among learners of all ages. In this article, we explore the impact of the global pandemic on a massive open online course, *Problem Solving Using Computational Thinking*, with a particular focus on the topics learners chose for their final projects. The *Computational Thinking* MOOC was designed using a project-based learning approach and aims to provide learners with an introduction to the "big ideas" of computational thinking using a range of case studies that encompass topics such as airport surveillance, epidemiology, and human trafficking. Beyond observing a sharp increase in enrollment and engagement at the time the pandemic began, we discuss ways in which the course's project-based pedagogy allowed learners to bring their present**

Digital Object Identifier 10.1109/MCSE.2020.3024012
Date of current version 9 October 2020.

experiences and concerns together with the course's subject matter in order to meet the learning objectives for the course. Many learners chose to address aspects of the pandemic in the course's final project and applied ideas about computational thinking to peer-graded assignments that conveyed an individualized sense of importance and urgency. We assert that this approach, along with the inclusion of a timely epidemiology case study, enabled learners to more deeply internalize the role that computational thinking can play in their own lives and in society as a whole.

The COVID-19 pandemic has thrust online learning environments into the global spotlight. The sudden shift to emergency remote teaching[1] has forced educators and students to adjust their educational routines using digital platforms and even pedagogical approaches with which they were previously unfamiliar.[2] Schools and universities have confronted this new reality, while lifelong learners have sought new online options for advancing their education.

One of the outcomes of these events has been an unprecedented rise in enrollments in massive open online courses (MOOCs), leading to a series of publications on the subject.[3] In this article, we explore the impacts of the pandemic on one MOOC: *Problem Solving Using Computational Thinking (CT)*. We give special attention to the educational benefits of a pedagogical approach that is rarely used in courses of this scale: project-based learning (PBL). We identify particular benefits of this approach that current events have highlighted, and we offer implications for teaching CT in online settings.

COURSE DESCRIPTION

Audience and Scope

The *CT* MOOC was developed by an interdisciplinary team from a large public university in the Midwestern United States. The course was designed for a target audience of precollege learners and early college learners who intend to pursue STEM careers and would thus need to develop fluency in the computational tools used in STEM. It aims to equip students with the modes of thinking needed to set up problems and potential solutions as a foundation for being able to eventually use computational tools and programming to address those problems.

Given this perspective, the course is based on one particular definition of CT. CT has been described and discussed in some form for decades[4], but it has received increased attention in recent years, with an especially sharp rise in relevant scholarship since 2015.[5] The definition of CT adopted in this course draws heavily on the work of Wing,[6] who presents CT as the practice of conceptualizing problems, complementing and combining mathematical and engineering thinking. Wing[6] argues that CT should not become a synonym for computer programming and notes that there are a range of skills necessary for CT, including the ability to define problems, reformulate seemingly intractable problems into solvable ones, use abstraction and decomposition when approaching a complex task, and use massive amounts of data and computation for problem solving.

While the *CT* MOOC similarly emphasizes the problem solving aspect and many of the skills that Wing highlights, it does so without requiring learners to make use of actual data or computer programming. In the course introduction, CT is defined in the following approachable way.

> *"Before you can think about programming a computer, you need to work out exactly what it is you want to tell the computer to do. Thinking through problems this way is CT. CT allows us to take complex problems, understand what the problem is, and develop solutions. We can present these solutions in a way that both computers and people can understand."*

The goals of the MOOC, therefore, focus on helping learners to specifically define and decompose problems through abstraction while teaching them to use insights from similar problems in other domains to guide potential solutions.

Project-Based Pedagogy

This MOOC was designed around a "project-based learning" (PBL) approach, which integrates instructional activities within projects motivated by students' own interests and contexts.[7] PBL is popular in K-12 classrooms, but there are questions of its utility in other settings, especially in MOOCs, which typically adopt a didactic, lecture-based format with instructional videos, quizzes, discussions, and graded assignments. However, some early work has found positive attitudes among learners in a project-based MOOC and has emphasized the importance of learner autonomy in these contexts.[8]

The *CT* MOOC centers much of its pedagogy around a final project in which learners identify a problem to solve *computationally*, and then use the knowledge and techniques they learned throughout the course to iteratively develop an algorithmic approach toward a solution. They are asked to submit both a graphic organizer that displays the multiple iterations of their work and a diagrammed algorithm of their final solution for peer evaluation. This approach affords learners a level of flexibility rarely found at the scale of MOOCs since they are able to select the topic that will define their final projects.

Using Case Studies to Shape Student Work

Using case studies is a way to ground CT in real-life scenarios. This provides a concrete, actionable foundation for CT that can more effectively lead to learning, retention, and application.[9] For this purpose, the *CT* course revolves around a series of three case studies.

Most pertinent here is the case on epidemiology, which was incidentally developed before the events surrounding the global pandemic. This case presents a large, complex problem: how do we prepare for the seasonal flu and make sure we are ready for the next pandemic? The expert who presents this case—an associate professor of epidemiology—breaks this down into a more specific subproblem and then walks learners through an algorithm that centers around four categories of people: vaccinated, susceptible, infected, and recovered. Finally, a series of computational modeling tools specific to epidemiology are offered to learners interested in diving more deeply into this problem.

ENROLLMENT AND ENGAGEMENT TRENDS

As a reflection of the recent spike in attention that online education has garnered, the *CT* MOOC experienced a sharp increase in enrollment around the time that many countries began to experience the full force of the COVID-19 pandemic. We found that most new learners who reported their educational attainment and employment status hold a bachelor's degree and are unemployed (see Figure 1). As expected, this suggests that the rise in overall enrollments is likely due to a combination of more people working from home, universities shifting to remote teaching, and rising unemployment rates.

Since MOOCs are notorious for having high numbers of enrollments with few learners actually engaging in the material or completing the course, we also explored learner engagement trends, calculated as the number of learners who completed at least one item (lecture video, quiz, discussion post, etc.) during the previous week. We discovered an even sharper rise coinciding with the spread of the virus, suggesting an impressive surge of learners actively engaged in course activities.

STUDENT FINAL PROJECTS

The course's peer-graded final assignment requires students to identify a location and natural disaster—either hypothetical or real—and create a preparation plan to address a specific aspect of this disaster. The plan must involve the main elements of CT described throughout the course, namely *problem identification*, *decomposition*, *pattern recognition*, and *abstraction*. Learners' final submissions consist of a descriptive problem statement, a graphic organizer, and an algorithm diagram. Examples of the wide range of disasters learners chose include floods (by far the most common), cyclones, earthquakes, tsunamis, tornadoes, and droughts.

With the circumstances surrounding the global pandemic influencing enrollment and engagement in the MOOC, we were interested in whether its impact would also be realized in topics learners chose for their final projects. Furthermore, we were interested in the specific problems that learners identified related to the pandemic which they deemed "solvable" using CT approaches.

Methods for Identifying Projects Related to the Pandemic

In order to accurately measure the number of final projects submitted and those related to the COVID-19 pandemic, we first used a series of Jupyter Notebooks to clean the data. We discarded

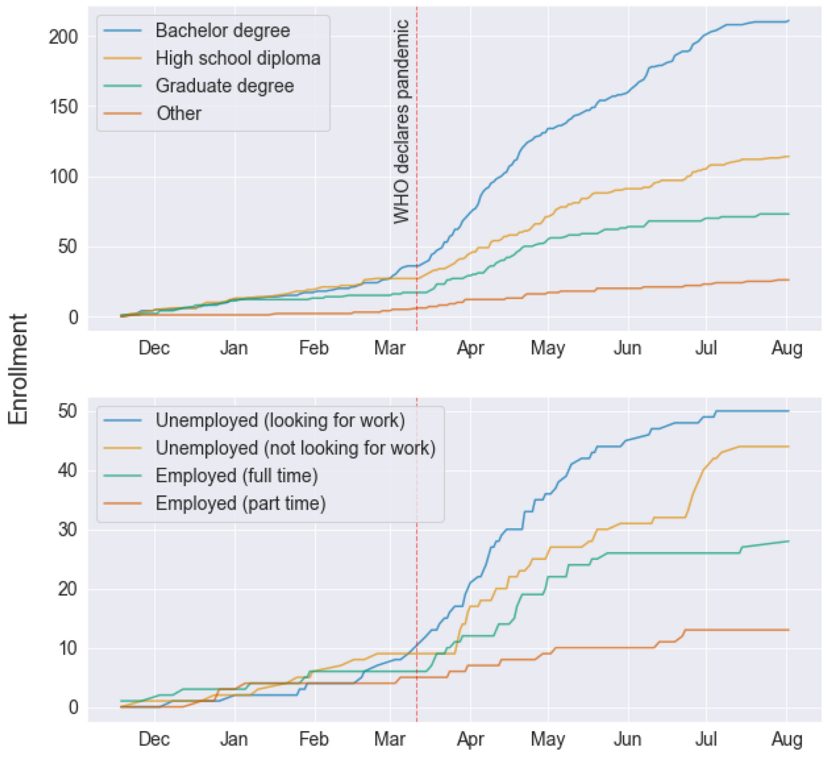


Figure 1. Enrollments by educational attainment (top) and employment status (bottom).

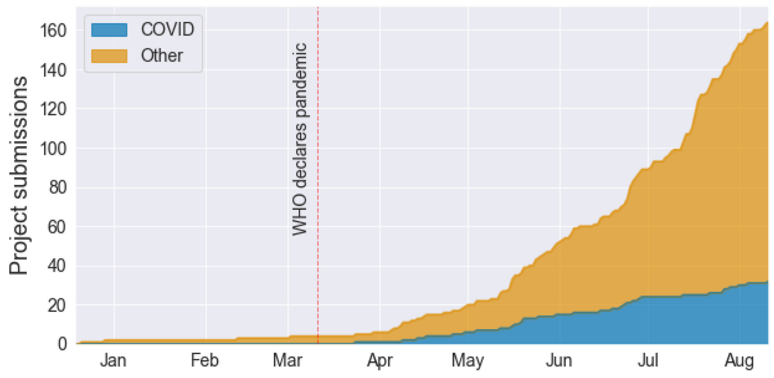


Figure 2. Proportion of pandemic-related final project submissions over time (stacked).

a total of 61 projects that were missing attachments, were exact duplicates of previously submitted projects, or skipped by peer reviewers due to incomplete submissions. This left us with 164 final projects to analyze, submitted between November 18, 2019 (course launch date) and August 11, 2020.

To identify projects related to the pandemic, we first extracted those that contained any of the following words in their title or description: *COVID*, *corona*, *virus*, *pandemic*, *epidemic*, *Wuhan*, and *social distanc(e)*. We then performed the same search through all PDF, DOCX, and PPTX project attachments. In total, we found 32 final projects that related to the events surrounding the pandemic.

Findings

The timing of pandemic-related final projects clearly corresponds with the rise in coronavirus cases outside of China, with the first related project submitted on March 24th and quickly increasing from there (see Figure 2). Since that date, 20% of submitted projects have been related to COVID-19 in some way.

We were interested in exploring whether learners in regions that were first heavily affected by the pandemic, such as East Asia or Europe, chose to write about the pandemic before others. However, we found that the first pandemic-related project was submitted in the United States, followed by India, Thailand, and then Germany, in that order (based on self-reported demographic data and tracked IP addresses).

As issues and problems surrounding the pandemic unfolded in real time, learners articulated specific dimensions that could be addressed using approaches advanced in the course. Some of the overarching topics in these projects include being wise about treatment, finding ways to prevent further spread, dealing with life in quarantine, managing school, and how to adequately distribute resources to those most affected. Interestingly, early projects were mostly about deciding to enforce stay-at-home orders and social distancing measures, while later projects largely focused on whether and when to begin reopening society.

Some learners who chose to write about the pandemic articulated that they recognized COVID-19 did not cleanly fit within the category of *natural disaster*.

> *"Rather than focusing on a "natural disaster" like too much snow or flooded streets, I have chosen COVID-19 since it has limited people's access to proper nutrition and healthy diet."*
>
> *"COVID-19 is a pandemic the whole world is talking about right now. This may not be a natural disaster but is more effective than a natural disaster."*

These excerpts—along with learners' decisions to choose a different path than instructed—highlight the weight of the pandemic in their personal lives.

IMPLICATIONS AND CONCLUSION

Our findings in this case study have highlighted the benefits of a PBL approach to teaching problem solving methods and skills such as CT. Though providing learners with a high degree of agency in online courses that are offered at scale may not always be feasible, doing so can empower them and help them establish personal ties to the subject matter. This heightened level of buy-in from learners, created through a combination of greater autonomy and personal interest in the subject, has been shown to lead to more robust learning.[10]

In the case of the *Problem Solving Using CT* MOOC, learners were able to choose project topics closely related to their interests, their experiences, and the large problems currently on their minds. For those who chose to look at the COVID-19 pandemic through a CT lens, we expect that the ongoing nature of the problem will lead to many additional opportunities to reflect on the skills they have learned. We assert that the PBL approach adopted in this course, along with the inclusion of a timely epidemiology case study, enabled learners to more deeply internalize the role that CT can play in their own lives and in society as a whole.

REFERENCES

1. C. Hodges, S. Moore, B. Lockee, T. Trust, and A. Bond, "The difference between emergency remote teaching and online learning," EDUCAUSE Rev., 2020. [Online]. Available: https://er.educause.edu/articles/2020/3/the-difference-between-emergency-remote-teaching-and-online-learning
2. R. Quintana and C. Quintana, "When classroom interactions have to go online: The move to specifications grading in a project-based design course," *Inf. Learn. Sci.*, vol. 121, no. 7/8, pp. 525–532, 2020, doi: 10.1108/ils-04-2020-0119.
3. S. Lohr, *Remember the MOOCs? After Near-Death, They're Booming*, New York, NY, USA: The New York Times, 2020. [Online]. Available: https://www.nytimes.com/2020/05/26/technology/moocs-online-learning.html
4. S. Papert, "An exploration in the space of mathematics educations," *Int. J. Comput. Math. Learn.*, vol. 1, no. 1, pp. 95–123, 1996, doi: 10.1007/bf00191473.
5. T.-C. Hsu, S.-C. Chang, and Y.-T. Hung, "How to learn and how to teach computational thinking: Suggestions based on a review of the literature," *Comput. Educ.*, vol. 126, pp. 296–310, 2018, doi: 10.1016/j.compedu.2018.07.004.
6. J. M. Wing, "Computational thinking," *Commun. ACM*, vol. 49, no. 3, pp. 33–35, 2006, doi: 10.1145/1118178.1118215.
7. R. K. Sawyer, "Project-based learning," in *The Cambridge Handbook of the Learning Sciences.* Cambridge, U.K.: Cambridge Univ. Press, 2005, pp. 317–334, doi: 10.1017/cbo9780511816833.020.

8. I. Levin and D. Tsybulsky, “Project-based MOOC: Enhancing knowledge construction and motivation to learn,” in *Digital Tools and Solutions for Inquiry-Based STEM Learning*. Hershey, PA, USA: IGI Global, 2017, pp. 282–307, doi: 10.4018/978-1-5225-2525-7.ch011.
9. D. Weintrop, “Defining computational thinking for mathematics and science classrooms,” *J. Sci. Educ. Technol.*, vol. 25, no. 1, pp. 127–147, 2015, doi: 10.1007/s10956-015-9581-5.
10. E. Patall, “Constructing motivation through choice interest, and interestingness,” *J. Educ. Psychol.*, vol. 105, no. 2, pp. 522–534, 2013, doi: 10.1037/a0030307.

Juan D. Pinto is currently a Learning Experience Design Research Fellow with the Center for Academic Innovation, University of Michigan, Ann Arbor, MI, USA, and is currently working toward a Doctoral degree with the University of Illinois at Urbana-Champaign, Champaign, IL, USA, studying digital environments for learning, teaching, and agency. His research interests include adaptive learning environments, educational data science, and remote digital learning. He received the M.A. degree in design and technologies for learning from the University of Michigan. Contact him at jdpinto2@illinois.edu.

Chris Quintana is currently an Associate Professor with the School of Education, University of Michigan, Ann Arbor, MI, USA, where he applies his background in human–computer interaction and computer science as he looks at how different technologies and media can support learning. Much of his work has focused on software-based scaffolding, including the development of scaffolded software tools, scaffolding frameworks, and learner-centered design processes. He received the Ph.D. degree from the University of Michigan in computer science and engineering. Contact him at quintana@umich.edu.

Rebecca M. Quintana is currently the Learning Experience Design Lead with the Center for Academic Innovation, University of Michigan, Ann Arbor, MI, USA, and an Intermittent Lecturer with the School of Education. In these roles she applies her background in learning sciences and educational technologies to consider how design and technology can support learning. Much of her work has focused on the design of online learning environments in higher education contexts. She received the Ph.D. degree from the Department of Curriculum, Teaching, and Learning, Ontario Institute for Studies in Education, University of Toronto, Toronto, ON, Canada. Contact her at rebeccaq@umich.edu.

Department: Computing Prescriptions
Editor: Francis Sullivan, fran@super.org
Reviewed by: Matthias Troyer

A Comparison of Quantum and Traditional Fourier Transform Computations

D. R. Musk
Pioneer Academics

***Abstract*—The quantum Fourier transform (QFT) can calculate the Fourier transform of a vector of size N with time complexity $\mathcal{O}(\log^2 N)$ as compared to the classical complexity of $\mathcal{O}(N \log N)$. However, if one wanted to measure the full output state, then the QFT complexity becomes $\mathcal{O}(N \log^2 N)$, thus losing its apparent advantage, indicating that the advantage is fully exploited for algorithms when only a limited number of samples is required from the output vector, as is the case in many quantum algorithms. Moreover, the computational complexity worsens if one considers the complexity of constructing the initial state. In this article, this issue is better illustrated by providing a concrete implementation of these algorithms and discussing their complexities as well as the complexity of the simulation of the QFT in MATLAB.**

Digital Object Identifier 10.1109/MCSE.2020.3023979
Date of publication 15 September 2020; date of current version 9 October 2020.

PRINCIPLES OF QUANTUM COMPUTING

Quantum computation poses a fundamentally different framework than our more familiar classical computation. Specifically, this is due to the existence of qubits, as opposed to bits, the fundamental components of computing. Whereas bits can only hold the binary values of 0 or 1, qubits can instead hold a superposition of states. We use Dirac's bra-ket notation, which utilizes two kets analogous to the classical states 0 and 1, to represent an orthogonal basis

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \text{ and } |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}. \tag{1}$$

A more generic qubit state is a superposition of the previous basis states

$$|\psi\rangle = x_0|0\rangle + x_1|1\rangle \tag{2}$$

where x_0 and x_1 are complex-valued amplitudes obeying the relation

$$|x_0|^2 + |x_1|^2 = 1. \tag{3}$$

It is important to clarify that such a superposition does not take a value "in between 0 and 1": instead, in accordance with the Born rule,[1] a measurement of the state of the qubit corresponds to a binary value. The qubit has a *probability* $|x_0|^2$ to be found in state "0" and a *probability* $|x_1|^2$ to be found in state "1."

We can use the Kronecker tensor product to construct the basis of a system comprised of multiple (n) qubits. For example, the product basis states of a four-dimensional ($n = 4$) linear vector space are

$$|\mathbf{0}\rangle = |0\rangle \otimes |0\rangle = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, |\mathbf{1}\rangle = |0\rangle \otimes |1\rangle = \begin{bmatrix}0\\1\\0\\0\end{bmatrix} \tag{4}$$

$$|\mathbf{2}\rangle = |1\rangle \otimes |0\rangle = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, |\mathbf{3}\rangle = |1\rangle \otimes |1\rangle = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}. \tag{5}$$

Given n qubits, the general state is a superposition state vector in a 2^n-dimensional Hilbert space

$$|X\rangle = \sum_{j=0}^{j<2^n} x_j|j\rangle \tag{6}$$

with $\sum_{j=0}^{j<2^n} |x_j|^2 = 1$.

Given an arbitrary state result of a computation, to measure one x coefficient j, one would have to perform a projection on the corresponding base vector

$$x_k = \langle k|x\rangle = \sum_{j=0}^{j<2^n} x_j\delta_{kj}. \tag{7}$$

δ_{kj} is the Kronecker delta. This measurement destroys the state and therefore only one coefficient (or one linear combination of coefficients) can be known. To know more coefficients, one would have to repeat the experiment that produced the original state. In other words, each measurement only provides one x_j of information, and one thus has to perform $\mathcal{O}(N)$ measurements to obtain all N coefficients to a certain precision.

This is a point of major importance that will affect our analysis of the quantum Fourier transform (QFT).

To compute with qubits, one can form quantum logic gates, which are analogous to classical logic gates but instead operate on quantum states. For example, the following gate H inputs state $|X\rangle$ and outputs $|Y\rangle$

$$|Y\rangle = H|X\rangle. \tag{8}$$

A more specific example would be the mapping of the basis state $|0\rangle$ to $(|0\rangle + |1\rangle)/\sqrt{2}$ and $|1\rangle$ to $(|0\rangle - |1\rangle)/\sqrt{2}$ which would be represented by the one-qubit Hadamard matrix

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}. \tag{9}$$

Another common example would be the controlled phase shift gate, a single-qubit phase shift gate that will leave the basis state $|0\rangle$ unchanged and map $|1\rangle$ to $\exp(i\phi)|1\rangle$, for a phase shift ϕ (such that the probability of measuring either a $|0\rangle$ or a $|1\rangle$ also remains unchanged).

In general, quantum gates are unitary operators that map the Hilbert space into itself. They are time evolution operators and they thus conserve energy and information. Moreover, although free quantum evolution requires no energy, true quantum gates instead require energy to operate, as one has to initiate and stop the interactions. Additionally, because quantum error corrections techniques make use of classical hardware, error correction is expected to be a major part of the energy budget in quantum computing.

DEFINITION AND USE OF THE CLASSICAL DISCRETE FOURIER TRANSFORM (DFT)

Before we move on to the first quantum algorithms, let us first examine the limitations of classical computation via one particular algorithm: the general number field sieve (GNFS).

The GNFS is currently the most efficient known classical algorithm for large integer factorization (particularly for integers exceeding 10^{100}).[2] A generalization of the special number field sieve, this algorithm works in subexponential time, specifically of complexity $\mathcal{O}[\exp(1.9(\log N)^{1/3}(\log\log N)^{2/3})]$.

However, we compare it to one of the first quantum algorithms to ever spark interest in the field of quantum computation: Shor's algorithm.[3] Via fast multiplication algorithms,[4] the algorithm need only take quantum gates of order $\mathcal{O}[(\log N)^2(\log\log N)(\log\log\log N)]$, thus being a member of the bounded-error quantum polynomial time complexity class.

One distinguishing feature of the Shor's algorithm is that it requires only a few repeated measurements of the output state to obtain the desired result such that one does not need to know all the coefficients of the output state. This is done via the quantum period finding subroutine, which does not load classical data and evaluate a full vector but instead finds the period by measuring the result of the QFT multiple times, which will be valuable for quantum speedup. In fact, basing factoring on period finding by using the QFT is the great innovation in Shor's quantum factoring algorithm.

In fact, there is an almost exponential difference between the complexities of the GNFS and Shor's algorithm, showing that quantum computing is, in principle, capable of performing tasks that no classical computer could ever perform: this phenomenon is often referred to as *quantum speedup*. To better visualize this complexity, Figure 1 graphs the number of operations as function of the input size.

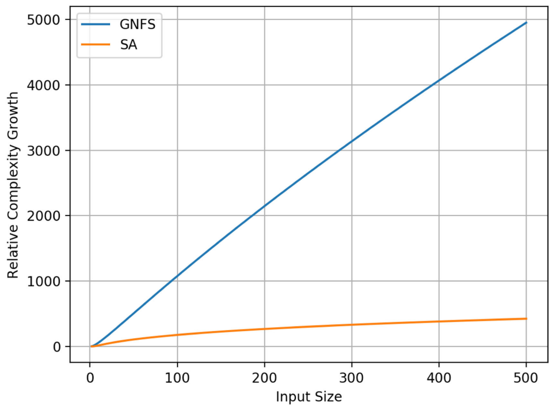


Figure 1. Scaling, represented by the number of operations as function of the algorithm's input size. (This graph was normalized to start at the origin, with SA denoting Shor's algorithm.)

LIMITATIONS OF CLASSICAL ALGORITHMS

Before introducing the QFT, for clarity, we define its classical counterpart: the DFT. The DFT maps a sequence of N complex numbers $\boldsymbol{x} = \{x_0, x_1, \ldots, x_{N-1}\}$ into another sequence, $\boldsymbol{y} = \{X_0, X_1, \ldots, X_{N-1}\}$, such that

$$y_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N}kn}, \quad k = 0, 1, 2, \ldots, N-1. \tag{10}$$

Moreover, an N-point DFT is often expressed as

$$\mathbf{y} = W\mathbf{x} \tag{11}$$

where W is the DFT matrix

$$W = \frac{1}{\sqrt{N}}\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}. \tag{12}$$

Here, $\omega = \exp(-2\pi i/N)$ is the primitive Nth root of unity.

This mathematical operation is typically implemented as an algorithm that minimizes the number of elementary arithmetic operations, which is known as the fast Fourier transform (FFT).[5] This reduces the complexity by directly computing the DFT from $\mathcal{O}(N^2)$ to $\mathcal{O}(N\log N)$.[6] Here, we will be using the Cooley–Tukey algorithm[7] to implement the FFT.

The simplest implementation of the Cooley–Tukey algorithm is the radix-2 decimation-in-time (DIT) case, which divides a DFT of size N into two interleaved DFTs with size $N/2$ for each stage in the algorithm's recursion; the next section examines an implementation of this method in MATLAB.

IMPLEMENTATION OF THE RADIX-2 DIT CASE IN MATLAB

The radix-2 DIT is a recursive algorithm, taking the following inputs: x, the relevant data, the input data x and its size $N = 2^n$, and the stride z (that is the distance in memory between consecutive values of x; we are taking an initial value of $z = 1$).

A possible implementation of the algorithm in MATLAB is as follows:

```
function d = DIT(x, N, z)
  if N == 1
    d(1) = x(1)
  else
    d(1:N/2) = DIT(x, N/2, 2*z)
    d(N/2+1:N) = DIT(x+z, N/2, 2*z)
    for k = 1:N/2
      t = d(k)
      d(k) =
       t + exp(-2*pi*i*(k-1)/N)*d(k+N/2)
      d(k+N/2) =
       t - exp(-2*pi*i*(k-1)/N)*d(k+N/2)
    end
  end
end
```

In the function's initial if statement, we account for the trivial case wherein x is a single element list; we thus set our output equal to the first member of x (which will, of course, always be equivalent to x).

In the else statement, we truly access the essence of the algorithm by first setting the first half of the output equal to the recursion of the function with a halved N and a doubled z (thus reiterating through the function until the elimination case is reached) and then setting the second half of the output to the shifted input by z with a halved N and a doubled z. Afterwards, we interleave the two evaluations to get our full DIT. This results in some interesting complexity analysis, which is reviewed in the following section.

EVALUATING THE COMPLEXITY ZOF THE RADIX-2 DIT IMPLEMENTATION

The runtime of the input of our implementation need only depend on the size of our input, N. Thus, we can describe this runtime via the usage of some unknown function, $T(N)$. As a result, the total runtime of our DIT implementation can be expressed as $T(N)$, and the recursive computations in line 5 and 6 as $T(N/2)$ (since we are merely reevaluating the function with a reduced N). Finally, the runtime of the computations present in the final for loop is proportional to N. However, since we are to evaluate the overall complexity of our algorithm, we can ignore any such scaling factors and simply write N. Therefore, we have

$$T(N) = 2T\left(\frac{N}{2}\right) + N. \tag{13}$$

Next, we can utilize the master theorem for divide-and-conquer recurrences for a more concrete answer. The master theorem states that for recurrence relations of the form $T(N) = aT(N/B) + f(N)$, we determine the critical exponent $c_{\text{crit}} = \log_b a$. Therefore, we utilize the case of the master theorem stating that $f(N) = \mathcal{O}(N^{c_{\text{crit}}} \log^k N)$ for any $k \geq 0$, since we merely have $c_{\text{crit}} = \log_2 2 = 1$ and we can thus simply set $k = 0$ such that $\log^k N = \log^0 N = 1$. Therefore, in accordance to the master theorem, we determine the function as having complexity

$$T(N) = \mathcal{O}(N^{c_{\text{crit}}} \log^{k+1} N) = \mathcal{O}(N \log N). \tag{14}$$

As expected, our implementation makes correct usage of the Cooley–Tukey algorithm, showing the aforementioned complexity.

DEFINITION AND USAGE OF QFT

Akin to the DFT that we examined in the section "Evaluating the Complexity of the radix-2 DIT Implementation," the QFT maps a quantum state $|x\rangle = \sum_{i=0}^{N-1} x_i |i\rangle$ to $\sum_{i=0}^{N-1} y_i |i\rangle$ in accordance to the formula

$$y_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n \omega^{nk}, \; k = 0, 1, 2, \ldots, N-1. \tag{15}$$

(We continue to use the notation of ω as a function of N: for further clarity, $\omega = \exp(-2\pi i/N)$ for $N = 2^n$.) Moreover, we can also re-express the QFT in a way that will be easier to simulate for our implementation,[8] which represents our input as a tensor product of states like so (here, s is used to better distinguish it from x)

$$|s\rangle = |s_1 s_2 \ldots s_n\rangle = |s_1\rangle \otimes |s_2\rangle \otimes \ldots \otimes |s_n\rangle. \tag{16}$$

We also borrow fractional binary notation such that

$$[0.s_1 \ldots s_m] = \sum_{k=1}^{m} s_k 2^{-k}. \tag{17}$$

Therefore, we can express the QFT as

$$\mathrm{QFT}(|s\rangle) = \frac{1}{\sqrt{2^n}} \bigotimes_{j=1}^{n} \left(|0\rangle + e^{2\pi i [0.s_j s_{j+1} \ldots s_n]}|1\rangle\right). \tag{18}$$

More precisely, (18) represents a time evolution operator constituted by the composition of n^2 gates in parallel, some Hadamard gates, and some controlled phase shift gates. Here, the larger tensor product symbol functions as an indexed product of the Kronecker tensor product. Via decomposition, the QFT on 2^n amplitudes can be implemented as a quantum circuit consisting of $\mathcal{O}(n^2)$ Hadamard gates and controlled phase shift gates for a number of qubits n.

The previously classical DFT would, in our implementation, require $\mathcal{O}(n2^n)$ gates, which is exponentially greater than the aforementioned quantum complexity. This may also be compared with $\mathcal{O}(n \log n)$, the number of gates required by the most efficient known QFT algorithms for an approximate result.

This analysis, however, does not take into account the complexity of preparing the input state and measuring the output states, which both have complexity $\mathcal{O}(N \log 1/\epsilon)$ for a required resolution ϵ.[9] It is therefore evident that in the case of the best-known QFT algorithm, the complexity is completely dominated by these measurements. Furthermore, if one wants to read out the full output vector the complexity becomes $\mathcal{O}(N^2 \log^2 1/\epsilon)$. The classical FFT algorithm also has a dependence on the desired precision, but we treat it as a constant depending on machine precision, not considering arbitrary precision implementations of the classical algorithm.

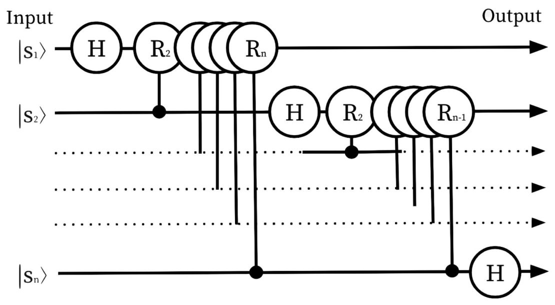


Figure 2. Quantum circuit drawn for the QFT for n qubits.

However, the complexity analysis of our simulation on a classical computer will not be fully accurate to the innate properties of a quantum device. On a classical computer, we cannot simulate the full QFT algorithm but we can simulate the application of the unary operator that corresponds to the QFT algorithm to one base state. We first examine a gate-level accurate implementation that we may write in pseudocode. As Figure 2 indicates, the QFT may be constructed with Hadamard gates (written H) and controlled phase shift gates (written R_m).

In pseudocode, we may write

```
function QFT(s):
  for i in 1..n:
    H(s_i)
    for m in 2{\ldots}{n-i+1}
      R_m(s_i)
```

where $H(..)$ is an application of the Hadamard gate and $R_m(\ldots)$ is an application of the controlled phase shift gate as in

$$R_m = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}$$

for some phase shift ϕ. They both may be thought of as operators acting in place on the state s_i.

The above algorithm consists of two nested loops therefore it has a complexity of $\mathcal{O}(n^2) = \mathcal{O}(\log^2 N)$.

Since each quantum gate can be simulated in $\mathcal{O}(N)$, and taking advantage of the possibility of parallelizing some of the transformations implemented by the gates, then we can simulate the above circuit in $\mathcal{O}(N \log^2 N)$.

We provide a possible implementation of this algorithm in MATLAB.[†] using the QUBIT4MATLAB[10] library.

EVALUATING THE COMPLEXITY OF THE QFT

In the previous section, we discussed the running time of the gate-accurate description of the QFT algorithm and proceeded to multiply by N (the cost of simulating the gate for each possible input) to obtain the gate-accurate simulation time.

We now discuss the complexity of (18) given one input s. One way to implement (18) consists of precomputing all required values of $[0.s_j \ldots s_n]$ (17) for each j. Since we have n possible values of j and the sum has a complexity proportional to n for each j, this operation has a total complexity of n^2.

We now consider the complexity of computing the Kronecker products in (18). This tensor product is effectively just a form of multiplication: using the associative property, we see that first product involves two vectors of size 2, resulting in a vector of size 4. The second product involves a vector of size 4 and a vector of size 2, resulting in a vector of size 8. This continues to the final result of size $N = 2^n$.

Thus, we effectively sum over 2^j as

$$\sum_{j=1}^{j<n} 2^j = \frac{2^n - 1}{2 - 1} = 2^n - 1 \in \mathcal{O}(2^n). \qquad (19)$$

To find the full complexity, we sum and take the greater term: $\mathcal{O}(n^2 + 2^n) = \mathcal{O}(2^n) = \mathcal{O}(N)$.

This possible implementation of the algorithm is faster than the gate accurate simulation previously discussed.

In principle, we take N basis vectors to account for arbitrary N coefficients, we should repeat these calculations N times. Yet, in practice, in simulation we are allowed to copy vectors therefore this does not affect the overall running time.

In evaluating the complexity of our function, we are mostly concerned with the complexity of the computations present in the repeated Kronecker product. We can compute its computational complexity in the simulation; however, this complexity does not appear in the (actual) quantum computer. We can represent the differences between the simulation and the true quantum computations as follows.

> We start with our initial state X, i.e., a superposition of base vectors. In the actual computation, the calculation of each component is parallelized at no cost but we can only measure one coefficient. We therefore need to rebuild the output state N times in order to be able to perform the N measurements needed to obtain all the coefficients.
> However, in the simulation, we must loop through the 2^n vector bases. As a result, the computation to generate the output state appears slower than the true quantum computation. Yet, because we can copy this state to perform multiple measurements, the net complexity is not slower. This apparent discrepancy is mostly due to the effect of the Kroneker delta computation.
> In true quantum information, we are only capable of extracting one measurement at a time. Thus, we compute every number 2^n faster than the solution. This forms a black box which computes every coefficient of the transform incredibly quickly: however, we may only measure one coefficient or linear combination of coefficients per each iteration of the function (since the QFT's wave function will collapse every time we call the black box, thus necessitating that the computations be redone).

In a true quantum computation, there is no cost in the Kronecker products discussed in the previous section. If the QFT was implemented as a series of Hadamard gates, this would require n^2 Hadamard gates (as was mentioned following the definition of the QFT in the "Definition and Usage of Quantum FOURIER Transform" section). Each of the $n^2 = \log^2 N$ gates require $O(N)$ classical operations to be simulated, thus, the total cost is $O(N \log^2 N)$.

[†]Source code of the algorithm in MATLAB may be found at https://github.com/DamianRMusk/MATLAB_FT

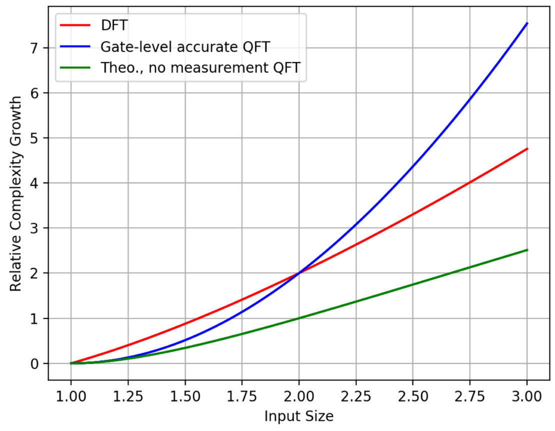


Figure 3. Our complexities in the form of a MATLAB graph, displaying the various forms of growth between the relative size of an algorithm's input and its overall complexity. (We have regularized some complexities to start at the origin.)

CONCLUSIONS

In Table 1, we can directly compare the formulated algorithmic complexities (to write terms in a uniform manner, we have reverted to expressing complexities in terms of $N = 2^n$ gates for the quantum algorithms).

We can graph the complexities as shown in Figure 3.

Accordingly, the complexities of our algorithms are rather distinct. The exponential (the DFT) increases far more rapidly than the other functions, the product of the input and the logarithm of the input (the simulated QFT) lies in-between the others, with the squared logarithm (the theoretical QFT) residing far below the two other functions, increasing at an almost linear rate: these visualized results further bring the concept of quantum speedup into question.

The previous results appear to confirm that true quantum algorithms and mere simulations of them have significantly different running times. In the case of the QFT, this is mostly due to the costliness of the iterated Kronecker tensor product for an arbitrary number of basis vectors. However, we arrive at the question of whether or not these complexities can truly be ranked directly. By the principles of quantum information, to acquire results more comparable to those given by the classical DFT, we must make extra preparations to our black box

Table 1. Our calculated complexities calculated by applying various complexity analysis techniques to our previously defined implementations. ("Sim" and "theo" are abbreviations for "simulated" and "theoretical," respectively.).

DFT	$\mathcal{O}(N\log N)$
QFT (gate-level accurate)	$\mathcal{O}(N\log^2 N)$
QFT (theo, no measurement)	$\mathcal{O}(\log^2 N)$
QFT (theo + preparation + measurement)	$\mathcal{O}(N^2\log^2 1/\epsilon)$

that, in turn, cost us a substantial amount of computing power (potentially indicating that, in actuality, the algorithms have comparable runtimes when evaluating the Fourier transforms in full).

More specifically, if we want to measure each coefficient, we must redo our operations for each coefficient (since our wave function will collapse for every measurement). This is not necessary for the classical DFT; upon making these modifications to the theoretical QFT such that it becomes more analogous to the DFT, it is possible that (in the case of Fourier analysis) quantum computing is no better than classical computing. However, this does not necessarily mean that quantum speedup is nonexistent; in the case of integer factorization, wherein we only desire a single answer that meets are conditions, the parallelization of quantum computing can become extremely useful (as mentioned in the Introduction in the case of Shor's algorithm, although this algorithm need only sample the wave function to calculate prime factors). In fact, this parallelization becomes beneficial for *any* type of search problem given the appropriate search criteria.[11]

The primary goal of this article was to present the reader with a nontrivial example to illustrate that directly comparing quantum algorithms to classical ones is a significant issue and there are a considerable number of potential hazards in doing so. Furthermore, in the case of some quantum algorithms, the perceived computational advantage may disappear should all elements of the wave function need to be read out, and there conversely exists speedup potential if only some elements need to be sampled. Therefore, although quantum speedup is certainly real for a number of algorithms, it is not necessarily a given.

ACKNOWLEDGMENTS

I would like to thank Professor Arthur Western for his guidance while working under him at Pioneer Academics. Moreover, I would like to acknowledge Lorena Barba and Matthias Troyer for their very detailed feedback and suggestions.[12]

REFERENCES

1. M. Born. *The Statistical Interpretation of Quantum Mechanics*. Amsterdam, The Netherlands: Elsevier, 1954.
2. C. Pomerance, "A tale of two sieves," *Notices Amer. Math. Soc.*, vol. 43, pp. 1473–1485, 1996.
3. P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, 1994, pp. 124–134.
4. D. Beckman, A. Chari, S. Devabhaktuni, and J. Preskill, "Efficient networks for quantum factoring," *Phys. Rev.*, vol. 54, pp. 1034–1063, 1996.
5. Y. Zhou, W. Cao, L. Liu, S. Agaian, and C. L. P. Chen, "Fast Fourier transform using matrix decomposition," *Inf. Sci.*, vol. 291, pp. 172–183, 1992.
6. M. Lohne, "The computational complexity of the fast Fourier transform," 2017.
7. A. Bekele, "Cooley–Tukey FFT algorithms," 2006.
8. V. H. Tellez, A. Campero, C. Iuga, and G. I. Duchen, "Quantum Fourier transform circuit simulator," in *Proc. Nano Sci. Technol. Inst. Nanotechnol. Conf. Trade Show*, 2008, pp. 39–42.
9. V. Shende, S. Bullock, and I. Markov, "Synthesis of quantum logic circuits," *IEEE Trans. Comput.-Aided Des.*, vol. 25, no. 6, pp. 1000–1010, Jun. 2006.
10. G. Toth, "Qubit4matlab v3.0: A program package for quantum information science and quantum optics for MATLAB," *Comput. Phys. Commun.*, vol. 179, pp. 430–437, 2008.
11. L. Grover, "Quantum computers can search rapidly by using almost any transformation," *Phys. Rev. Lett.*, vol. 80, pp. 4329–4332, 1997.
12. M. Troyer, "Review: A comparison of quantum and traditional Fourier transform computations," 2020, doi: 10.22541/au.159795474.47457948.

D. R. Musk is a student in Stanford Online High School and interns as a software developer at SpaceX. He participated in the Pioneer Academics research program under the research concentration "Fourier Series and Transforms with Applications in Physics and Related Fields," advised by Professor Arthur Western. His main interests are computational physics, theoretical physics, and discrete mathematics; he aspires to earn a Ph.D. in a related specialization. Contact him at damianmusk@gmail.com.

SUBMIT TODAY

IEEE TRANSACTIONS ON BIG DATA

SCOPE

The *IEEE Transactions on Big Data* (*TBD*) publishes peer reviewed articles with big data as the main focus. The articles provide cross disciplinary innovative research ideas and applications results for big data including novel theory, algorithms and applications. Research areas for big data include, but are not restricted to, big data analytics, big data visualization, big data curation and management, big data semantics, big data infrastructure, big data standards, big data performance analyses, intelligence from big data, scientific discovery from big data security, privacy, and legal issues specific to big data. Applications of big data in the fields of endeavor where massive data is generated are of particular interest.

SUBSCRIBE AND SUBMIT

For more information on paper submission, featured articles, calls for papers, and subscription links visit:

www.computer.org/tbd

Errata

Corrections to “An Exploration of Black Students Interacting With Computing College and Career Readiness Vlog Commentary Social Media Influencers”

Robert T. Cummings
Morehouse College

Earl W. Huff
Clemson University

Naja A. Mack
University of Florida

Kevin Womack
Morehouse College

Amber Reid
Clark Atlanta University

Brandon Ghoram
Morehouse College

Kinnis Gosha
Morehouse College

Juan E. Gilbert
University of Florida

In the above article,[1] Earl H. Huff should have been listed as Earl W. Huff. The correct author list is shown above.

Digital Object Identifier 10.1109/MCSE.2020.3023864
Date of current version 9 October 2020.

REFERENCES

1. R. Cummings *et al.*, “An exploration of Black students interacting with computing college and career readiness Vlog commentary social media influencers,” *Comput. Sci. Eng.*, vol. 22, no. 5, pp. 29–40, Sep./Oct. 2020.

Evolving Career Opportunities Need Your Skills

Explore new options—upload your resume today

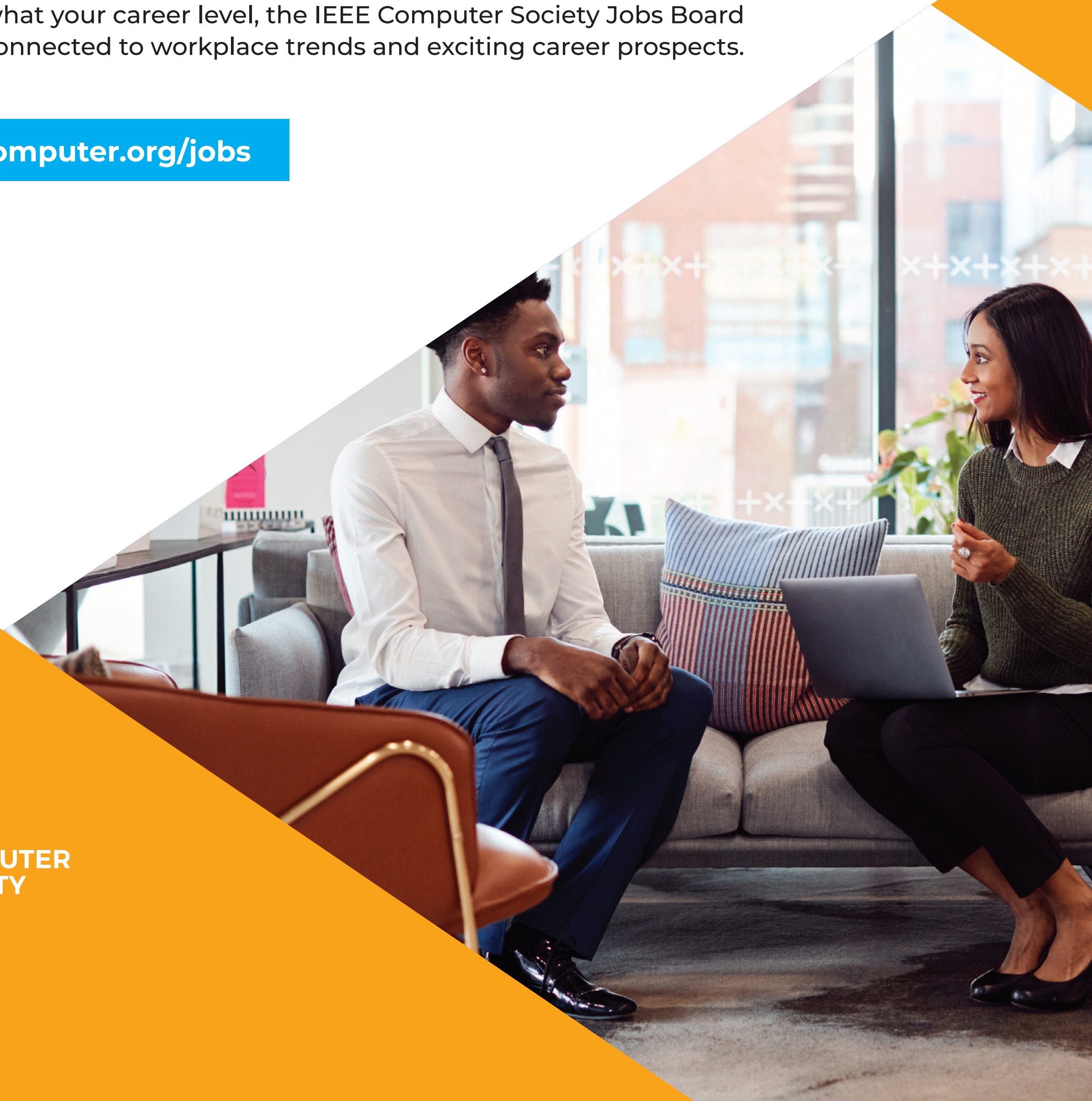
Changes in the marketplace shift demands for vital skills and talent. The **IEEE Computer Society Jobs Board** is a valuable resource tool to keep job seekers up to date on the dynamic career opportunities offered by employers.

Take advantage of these special resources for job seekers:

- JOB ALERTS
- TEMPLATES
- WEBINARS
- CAREER ADVICE
- RESUMES VIEWED BY TOP EMPLOYERS

No matter what your career level, the IEEE Computer Society Jobs Board keeps you connected to workplace trends and exciting career prospects.

www.computer.org/jobs



IEEE COMPUTER SOCIETY

IEEE